

FflasFfpack

Generated on Thu Dec 12 2024 00:00:00 for FflasFfpack by Doxygen 1.12.0

Thu Dec 12 2024 00:00:00



<b>1 FFLAS-FFPACK Documentation.</b>	<b>1</b>
1.1 Introduction . . . . .	1
1.2 Goals . . . . .	1
1.3 Design . . . . .	1
1.4 Using FFLAS-FFPACK. . . . .	1
1.5 Contributing to fflas-ffpack, getting assistance. . . . .	2
1.6 Copying and Licence . . . . .	2
1.7 Tutorial . . . . .	2
1.8 Configuring and Installing FFLAS-FFPACK . . . . .	2
1.9 Architecture of the library. . . . .	2
<b>2 Bug List</b>	<b>3</b>
<b>3 Bibliography</b>	<b>7</b>
<b>4 Todo List</b>	<b>9</b>
<b>5 Topic Index</b>	<b>13</b>
5.1 Topics . . . . .	13
<b>6 Namespace Index</b>	<b>15</b>
6.1 Namespace List . . . . .	15
<b>7 Hierarchical Index</b>	<b>17</b>
7.1 Class Hierarchy . . . . .	17
<b>8 Data Structure Index</b>	<b>25</b>
8.1 Data Structures . . . . .	25
<b>9 File Index</b>	<b>33</b>
9.1 File List . . . . .	33
<b>10 Topic Documentation</b>	<b>41</b>
10.1 CHECKER . . . . .	41
10.2 FFLAS-FFPACK . . . . .	41
10.2.1 Detailed Description . . . . .	41
10.2.2 FFLAS . . . . .	41
10.2.3 Interfaces . . . . .	42
10.3 Matrix Multiplication Algorithms . . . . .	42
10.3.1 Detailed Description . . . . .	42
10.4 SIMD wrapper . . . . .	42
10.5 FFPACK . . . . .	42
10.5.1 Detailed Description . . . . .	42
10.6 FFLAS-FFPACK fields . . . . .	43
10.6.1 Detailed Description . . . . .	43
10.7 RNS . . . . .	43

<b>11 Namespace Documentation</b>	<b>45</b>
11.1 FFLAS Namespace Reference	45
11.1.1 Typedef Documentation	72
11.1.1.1 Checker_fgemm	72
11.1.1.2 Checker_ftrsm	72
11.1.1.3 ForceCheck_fgemm	72
11.1.1.4 ForceCheck_ftrsm	72
11.1.1.5 ZOSparseMatrix	72
11.1.1.6 NotZOSparseMatrix	72
11.1.1.7 SimdSparseMatrix	73
11.1.1.8 NoSimdSparseMatrix	73
11.1.1.9 MKLSparseMatrixFormat	73
11.1.1.10 NotMKLSparseMatrixFormat	73
11.1.1.11 has_plus	73
11.1.1.12 has_minus	73
11.1.1.13 has_equal	73
11.1.1.14 has_plus_eq	73
11.1.1.15 has_minus_eq	73
11.1.1.16 has_mul	74
11.1.1.17 has_mul_eq	74
11.1.1.18 Timer	74
11.1.1.19 BaseTimer	74
11.1.1.20 UserTimer	74
11.1.1.21 SysTimer	74
11.1.2 Enumeration Type Documentation	74
11.1.2.1 FFLAS_ORDER	74
11.1.2.2 FFLAS_TRANSPOSE	74
11.1.2.3 FFLAS_UPLO	75
11.1.2.4 FFLAS_DIAG	75
11.1.2.5 FFLAS_SIDE	75
11.1.2.6 FFLAS_BASE	75
11.1.2.7 number_kind	76
11.1.2.8 SparseMatrix_t	76
11.1.2.9 FFLAS_FORMAT	76
11.1.3 Function Documentation	77
11.1.3.1 InfNorm()	77
11.1.3.2 min3()	77
11.1.3.3 max3()	77
11.1.3.4 min4()	77
11.1.3.5 max4()	77
11.1.3.6 fadd() [1/8]	78
11.1.3.7 faddin() [1/5]	78

11.1.3.8 fsub() [1/4]	78
11.1.3.9 fsubin() [1/3]	78
11.1.3.10 fadd() [2/8]	79
11.1.3.11 pfadd()	79
11.1.3.12 pfsb() [2/8]	79
11.1.3.13 pfaddin()	80
11.1.3.14 pfsbin()	80
11.1.3.15 fadd() [3/8]	80
11.1.3.16 fsub() [2/4]	81
11.1.3.17 faddin() [2/5]	81
11.1.3.18 faddin() [3/5]	81
11.1.3.19 fsubin() [2/3]	82
11.1.3.20 fadd() [4/8]	82
11.1.3.21 fassign() [1/10]	82
11.1.3.22 fassign() [2/10]	83
11.1.3.23 fassign() [3/10]	83
11.1.3.24 fassign() [4/10]	83
11.1.3.25 fassign() [5/10]	84
11.1.3.26 fassign() [6/10]	84
11.1.3.27 fassign() [7/10]	84
11.1.3.28 fassign() [8/10]	84
11.1.3.29 faxpy() [1/6]	85
11.1.3.30 faxpy() [2/6]	85
11.1.3.31 faxpy() [3/6]	85
11.1.3.32 faxpy() [4/6]	86
11.1.3.33 fdot() [1/11]	86
11.1.3.34 fdot() [2/11]	86
11.1.3.35 fdot() [3/11]	87
11.1.3.36 fdot() [4/11]	87
11.1.3.37 fdot() [5/11]	87
11.1.3.38 fdot() [6/11]	87
11.1.3.39 fdot() [7/11]	88
11.1.3.40 fdot() [8/11]	88
11.1.3.41 fgemm() [1/23]	88
11.1.3.42 fgemm() [2/23]	89
11.1.3.43 fgemm() [3/23]	89
11.1.3.44 fgemm() [4/23]	89
11.1.3.45 fgemm() [5/23]	90
11.1.3.46 fgemm() [6/23]	91
11.1.3.47 fsquare() [1/6]	91
11.1.3.48 fsquare() [2/6]	92
11.1.3.49 fsquare() [3/6]	92

11.1.3.50 fsquare() [4/6]	92
11.1.3.51 fsquare() [5/6]	92
11.1.3.52 fgemm() [7/23]	93
11.1.3.53 fgemm() [8/23]	93
11.1.3.54 fgemm() [9/23]	93
11.1.3.55 fgemm() [10/23]	94
11.1.3.56 fgemm() [11/23]	94
11.1.3.57 fgemm() [12/23]	95
11.1.3.58 fgemm() [13/23]	95
11.1.3.59 fgemm() [14/23]	95
11.1.3.60 fgemm() [15/23]	96
11.1.3.61 fgemm() [16/23]	96
11.1.3.62 fgemm() [17/23]	96
11.1.3.63 fgemm() [18/23]	97
11.1.3.64 fgemv() [1/19]	97
11.1.3.65 fgemv() [2/19]	98
11.1.3.66 fgemv() [3/19]	98
11.1.3.67 fgemv() [4/19]	98
11.1.3.68 fgemv() [5/19]	99
11.1.3.69 fgemv() [6/19]	99
11.1.3.70 fgemv() [7/19]	100
11.1.3.71 fgemv() [8/19]	100
11.1.3.72 fgemv() [9/19]	100
11.1.3.73 fgemv() [10/19]	101
11.1.3.74 fgemv() [11/19]	101
11.1.3.75 fgemv() [12/19]	101
11.1.3.76 fgemv() [13/19]	102
11.1.3.77 fgemv() [14/19]	102
11.1.3.78 fgemv() [15/19]	102
11.1.3.79 fgemv() [16/19]	103
11.1.3.80 fger() [1/12]	103
11.1.3.81 fger() [2/12]	104
11.1.3.82 fger() [3/12]	104
11.1.3.83 fger() [4/12]	104
11.1.3.84 fger() [5/12]	105
11.1.3.85 fger() [6/12]	105
11.1.3.86 fger() [7/12]	105
11.1.3.87 fger() [8/12]	106
11.1.3.88 fger() [9/12]	106
11.1.3.89 fger() [10/12]	106
11.1.3.90 fger() [11/12]	107
11.1.3.91 freduce() [1/11]	107

11.1.3.92 <code>freduce()</code> [2/11]	107
11.1.3.93 <code>freduce_constoverride()</code> [1/2]	108
11.1.3.94 <code>finit()</code> [1/8]	108
11.1.3.95 <code>finit()</code> [2/8]	108
11.1.3.96 <code>freduce()</code> [3/11]	109
11.1.3.97 <code>freduce()</code> [4/11]	109
11.1.3.98 <code>pfreduce()</code>	109
11.1.3.99 <code>freduce()</code> [5/11]	109
11.1.3.100 <code>freduce_constoverride()</code> [2/2]	110
11.1.3.101 <code>finit()</code> [3/8]	110
11.1.3.102 <code>finit()</code> [4/8]	110
11.1.3.103 <code>freduce()</code> [6/11]	111
11.1.3.104 <code>freduce()</code> [7/11]	111
11.1.3.105 <code>freivalds()</code>	111
11.1.3.106 <code>fscalin()</code> [1/10]	112
11.1.3.107 <code>fscal()</code> [1/10]	112
11.1.3.108 <code>fscal()</code> [2/10]	113
11.1.3.109 <code>fscal()</code> [3/10]	113
11.1.3.110 <code>fscalin()</code> [2/10]	113
11.1.3.111 <code>fscalin()</code> [3/10]	113
11.1.3.112 <code>fscalin()</code> [4/10]	113
11.1.3.113 <code>fscal()</code> [4/10]	114
11.1.3.114 <code>fscalin()</code> [5/10]	114
11.1.3.115 <code>fscal()</code> [5/10]	115
11.1.3.116 <code>fscalin()</code> [6/10]	115
11.1.3.117 <code>fscal()</code> [6/10]	115
11.1.3.118 <code>fscalin()</code> [7/10]	115
11.1.3.119 <code>fscal()</code> [7/10]	116
11.1.3.120 <code>fscalin()</code> [8/10]	116
11.1.3.121 <code>fscal()</code> [8/10]	116
11.1.3.122 <code>fsyr2k()</code>	116
11.1.3.123 <code>fsyrk()</code> [1/16]	117
11.1.3.124 <code>fsyrk()</code> [2/16]	118
11.1.3.125 <code>fsyrk()</code> [3/16]	118
11.1.3.126 <code>fsyrk()</code> [4/16]	118
11.1.3.127 <code>fsyrk()</code> [5/16]	119
11.1.3.128 <code>fsyrk()</code> [6/16]	119
11.1.3.129 <code>fsyrk()</code> [7/16]	119
11.1.3.130 <code>fsyrk()</code> [8/16]	120
11.1.3.131 <code>fsyrk()</code> [9/16]	120
11.1.3.132 <code>fsyrk()</code> [10/16]	120
11.1.3.133 <code>fsyrk()</code> [11/16]	121

11.1.3.134 fsyrk() [12/16]	121
11.1.3.135 fsyrk() [13/16]	122
11.1.3.136 fsyrk() [14/16]	122
11.1.3.137 computeS1S2()	123
11.1.3.138 fsyrk() [15/16]	123
11.1.3.139 fsyrk() [16/16]	124
11.1.3.140 fsyrk_strassen() [1/2]	124
11.1.3.141 ftrmm() [1/3]	124
11.1.3.142 ftrmm() [2/3]	125
11.1.3.143 ftrsm() [1/9]	126
11.1.3.144 ftrsm() [2/9]	126
11.1.3.145 ftrsm() [3/9]	127
11.1.3.146 ftrsm() [4/9]	127
11.1.3.147 ftrsm() [5/9]	127
11.1.3.148 cblas_imptrsm()	128
11.1.3.149 ftrsv() [1/2]	128
11.1.3.150 igemm_()	128
11.1.3.151 finit() [5/8]	129
11.1.3.152 fconvert() [1/3]	129
11.1.3.153 fnegin() [1/4]	130
11.1.3.154 fneg() [1/4]	130
11.1.3.155 fzero() [1/5]	131
11.1.3.156 frand() [1/2]	131
11.1.3.157 fiszero() [1/4]	131
11.1.3.158 fequal() [1/4]	132
11.1.3.159 faxpby() [1/2]	132
11.1.3.160 fdot() [9/11]	133
11.1.3.161 fswap() [1/2]	133
11.1.3.162 fzero() [2/5]	134
11.1.3.163 fzero() [3/5]	135
11.1.3.164 frand() [2/2]	135
11.1.3.165 fequal() [2/4]	136
11.1.3.166 fiszero() [2/4]	136
11.1.3.167 fidentity() [1/4]	137
11.1.3.168 fidentity() [2/4]	137
11.1.3.169 finit() [6/8]	137
11.1.3.170 fconvert() [2/3]	137
11.1.3.171 fnegin() [2/4]	138
11.1.3.172 fneg() [2/4]	138
11.1.3.173 faxpby() [2/2]	139
11.1.3.174 fmove() [1/2]	139
11.1.3.175 bitsize()	140



11.1.3.176 <code>bitsize&lt; Givaro::ZRing&lt; Givaro::Integer &gt; &gt;()</code>	140
11.1.3.177 <code>ftrmv()</code>	141
11.1.3.178 <code>ftism()</code> [6/9]	141
11.1.3.179 <code>fsyrk_strassen()</code> [2/2]	142
11.1.3.180 <code>pfgemm()</code> [1/7]	142
11.1.3.181 <code>pfgemm_1D_rec()</code>	143
11.1.3.182 <code>pfgemm_2D_rec()</code>	143
11.1.3.183 <code>pfgemm_3D_rec()</code>	143
11.1.3.184 <code>pfgemm_3D_rec2()</code>	144
11.1.3.185 <code>fgemm()</code> [19/23]	144
11.1.3.186 <code>ftism()</code> [7/9]	144
11.1.3.187 <code>ftism()</code> [8/9]	145
11.1.3.188 <code>fspmv()</code> [1/2]	145
11.1.3.189 <code>fspmm()</code>	145
11.1.3.190 <code>sparse_init()</code> [1/16]	146
11.1.3.191 <code>sparse_init()</code> [2/16]	146
11.1.3.192 <code>sparse_delete()</code> [1/12]	146
11.1.3.193 <code>sparse_delete()</code> [2/12]	146
11.1.3.194 <code>sparse_init()</code> [3/16]	146
11.1.3.195 <code>sparse_init()</code> [4/16]	147
11.1.3.196 <code>sparse_delete()</code> [3/12]	147
11.1.3.197 <code>sparse_delete()</code> [4/12]	147
11.1.3.198 <code>sparse_print()</code> [1/3]	147
11.1.3.199 <code>sparse_init()</code> [5/16]	147
11.1.3.200 <code>sparse_init()</code> [6/16]	148
11.1.3.201 <code>sparse_init()</code> [7/16]	148
11.1.3.202 <code>sparse_init()</code> [8/16]	148
11.1.3.203 <code>sparse_delete()</code> [5/12]	148
11.1.3.204 <code>sparse_init()</code> [9/16]	149
11.1.3.205 <code>sparse_init()</code> [10/16]	149
11.1.3.206 <code>sparse_init()</code> [11/16]	149
11.1.3.207 <code>sparse_delete()</code> [6/12]	149
11.1.3.208 <code>sparse_delete()</code> [7/12]	149
11.1.3.209 <code>sparse_init()</code> [12/16]	150
11.1.3.210 <code>sparse_init()</code> [13/16]	150
11.1.3.211 <code>sparse_delete()</code> [8/12]	150
11.1.3.212 <code>sparse_delete()</code> [9/12]	150
11.1.3.213 <code>sparse_print()</code> [2/3]	150
11.1.3.214 <code>sparse_delete()</code> [10/12]	150
11.1.3.215 <code>sparse_init()</code> [14/16]	151
11.1.3.216 <code>operator&lt;&lt;()</code>	151
11.1.3.217 <code>readSmsFormat()</code>	151

11.1.3.218 readSprFormat()	151
11.1.3.219 getDataType() [1/4]	151
11.1.3.220 getDataType() [2/4]	152
11.1.3.221 getDataType() [3/4]	152
11.1.3.222 getDataType() [4/4]	152
11.1.3.223 readMachineType()	152
11.1.3.224 readDnsFormat()	152
11.1.3.225 writeDnsFormat()	152
11.1.3.226 fspmv() [2/2]	153
11.1.3.227 sparse_delete() [11/12]	153
11.1.3.228 sparse_delete() [12/12]	153
11.1.3.229 sparse_print() [3/3]	153
11.1.3.230 sparse_init() [15/16]	153
11.1.3.231 sparse_init() [16/16]	153
11.1.3.232 computeDeviation()	154
11.1.3.233 getStat()	154
11.1.3.234 maxCardinality()	154
11.1.3.235 maxCardinality< Givaro::Modular< int64_t > >()	154
11.1.3.236 maxCardinality< Givaro::Modular< int32_t > >()	154
11.1.3.237 minCardinality()	154
11.1.3.238 fflas_delete() [1/4]	154
11.1.3.239 fflas_delete() [2/4]	155
11.1.3.240 fflas_new() [1/7]	155
11.1.3.241 fflas_new() [2/7]	155
11.1.3.242 finit_rns() [1/2]	155
11.1.3.243 finit_trans_rns()	155
11.1.3.244 fconvert_rns() [1/2]	156
11.1.3.245 fconvert_trans_rns()	156
11.1.3.246 fflas_new() [3/7]	156
11.1.3.247 fflas_new() [4/7]	156
11.1.3.248 finit_rns() [2/2]	156
11.1.3.249 fconvert_rns() [2/2]	157
11.1.3.250 freduce() [8/11]	157
11.1.3.251 freduce() [9/11]	157
11.1.3.252 finit() [7/8]	158
11.1.3.253 fconvert() [3/3]	158
11.1.3.254 fnegin() [3/4]	158
11.1.3.255 fneg() [3/4]	159
11.1.3.256 fzero() [4/5]	159
11.1.3.257 fiszero() [3/4]	160
11.1.3.258 fequal() [3/4]	160
11.1.3.259 fassign() [9/10]	160

11.1.3.260 fscaln()	[ 9/10]	161
11.1.3.261 fscal()	[ 9/10]	161
11.1.3.262 faxpy()	[ 5/6]	162
11.1.3.263 fdot()	[10/11]	162
11.1.3.264 fswap()	[ 2/2]	163
11.1.3.265 fadd()	[ 5/8]	164
11.1.3.266 fsub()	[ 3/4]	164
11.1.3.267 faddin()	[ 4/5]	164
11.1.3.268 fadd()	[ 6/8]	164
11.1.3.269 fassign()	[10/10]	165
11.1.3.270 fzero()	[ 5/5]	165
11.1.3.271 fequal()	[ 4/4]	166
11.1.3.272 fiszero()	[ 4/4]	166
11.1.3.273 fidentity()	[ 3/4]	166
11.1.3.274 fidentity()	[ 4/4]	167
11.1.3.275 freduce()	[10/11]	167
11.1.3.276 freduce()	[11/11]	167
11.1.3.277 finit()	[ 8/8]	168
11.1.3.278 fnegin()	[ 4/4]	168
11.1.3.279 fneg()	[ 4/4]	168
11.1.3.280 fscaln()	[10/10]	169
11.1.3.281 fscal()	[10/10]	169
11.1.3.282 faxpy()	[ 6/6]	170
11.1.3.283 fmove()	[ 2/2]	170
11.1.3.284 fadd()	[ 7/8]	171
11.1.3.285 fsub()	[ 4/4]	171
11.1.3.286 fsubin()	[ 3/3]	172
11.1.3.287 fadd()	[ 8/8]	172
11.1.3.288 faddin()	[ 5/5]	173
11.1.3.289 fgemv()	[17/19]	173
11.1.3.290 fger()	[12/12]	174
11.1.3.291 ftrsv()	[ 2/2]	174
11.1.3.292 ftrsm()	[ 9/9]	175
11.1.3.293 ftrmm()	[ 3/3]	176
11.1.3.294 fgemm()	[20/23]	176
11.1.3.295 fgemm()	[21/23]	177
11.1.3.296 fgemm()	[22/23]	177
11.1.3.297 fgemm()	[23/23]	178
11.1.3.298 fsquare()	[ 6/6]	178
11.1.3.299 BlockCuts()	[ 1/2]	179
11.1.3.300 BlockCuts< CuttingStrategy::Single, StrategyParameter::Threads >()		179
11.1.3.301 BlockCuts< CuttingStrategy::Row, StrategyParameter::Fixed >()		179

11.1.3.302 BlockCuts< CuttingStrategy::Row, StrategyParameter::Grain >()	179
11.1.3.303 BlockCuts< CuttingStrategy::Block, StrategyParameter::Grain >()	180
11.1.3.304 BlockCuts< CuttingStrategy::Column, StrategyParameter::Fixed >()	180
11.1.3.305 BlockCuts< CuttingStrategy::Column, StrategyParameter::Grain >()	180
11.1.3.306 BlockCuts< CuttingStrategy::Block, StrategyParameter::Fixed >()	180
11.1.3.307 BlockCuts< CuttingStrategy::Row, StrategyParameter::Threads >()	180
11.1.3.308 BlockCuts< CuttingStrategy::Column, StrategyParameter::Threads >()	181
11.1.3.309 BlockCuts< CuttingStrategy::Block, StrategyParameter::Threads >()	181
11.1.3.310 BlockCuts() [2/2]	181
11.1.3.311 pfzero()	181
11.1.3.312 pfrand()	181
11.1.3.313 fdot() [11/11]	182
11.1.3.314 pfgemm() [2/7]	182
11.1.3.315 pfgemm() [3/7]	182
11.1.3.316 pfgemm() [4/7]	183
11.1.3.317 pfgemm() [5/7]	183
11.1.3.318 pfgemm() [6/7]	183
11.1.3.319 pfgemm() [7/7]	184
11.1.3.320 fgemv() [18/19]	184
11.1.3.321 fgemv() [19/19]	184
11.1.3.322 parseArguments()	185
11.1.3.323 getArgumentValue()	185
11.1.3.324 writeCommandString()	185
11.1.3.325 WriteMatrix() [1/2]	186
11.1.3.326 preamble()	186
11.1.3.327 ReadMatrix() [1/2]	186
11.1.3.328 ReadMatrix() [2/2]	187
11.1.3.329 WriteMatrix() [2/2]	187
11.1.3.330 WritePermutation()	187
11.1.3.331 alignable()	188
11.1.3.332 alignable< Givaro::Integer * >()	188
11.1.3.333 fflas_new() [5/7]	188
11.1.3.334 fflas_new() [6/7]	188
11.1.3.335 fflas_new() [7/7]	188
11.1.3.336 fflas_delete() [3/4]	188
11.1.3.337 fflas_delete() [4/4]	189
11.1.3.338 prefetch()	189
11.1.3.339 getTLBSize()	189
11.1.3.340 queryCacheSizes()	189
11.1.3.341 queryL1CacheSize()	189
11.1.3.342 queryTopLevelCacheSize()	189
11.1.3.343 getSeed()	189

11.2 FFLAS::_ftranspose_impl Namespace Reference	190
11.2.1 Function Documentation	190
11.2.1.1 not_inplace()	190
11.2.1.2 square_inplace()	190
11.2.1.3 nonsquare_inplace_v1()	190
11.2.1.4 nonsquare_inplace_v2()	191
11.3 FFLAS::BLAS3 Namespace Reference	191
11.3.1 Function Documentation	192
11.3.1.1 Bini()	192
11.3.1.2 WinoPar()	193
11.3.1.3 Winograd()	193
11.3.1.4 WinogradAcc_3_23()	193
11.3.1.5 WinogradAcc_3_21()	194
11.3.1.6 WinogradAcc_2_24()	194
11.3.1.7 WinogradAcc_2_27()	195
11.3.1.8 WinogradAcc_LR()	195
11.3.1.9 WinogradAcc_R_S()	195
11.3.1.10 WinogradAcc_L_S()	196
11.3.1.11 Winograd_LR_S()	196
11.3.1.12 Winograd_L_S()	196
11.3.1.13 Winograd_R_S()	197
11.4 FFLAS::csr_hyb_details Namespace Reference	197
11.5 FFLAS::CuttingStrategy Namespace Reference	197
11.5.1 Typedef Documentation	198
11.5.1.1 RNSModulus	198
11.6 FFLAS::details Namespace Reference	198
11.6.1 Function Documentation	199
11.6.1.1 fadd() [1/5]	199
11.6.1.2 fadd() [2/5]	200
11.6.1.3 fadd() [3/5]	200
11.6.1.4 fadd() [4/5]	200
11.6.1.5 fadd() [5/5]	201
11.6.1.6 faxpy() [1/2]	201
11.6.1.7 faxpy() [2/2]	201
11.6.1.8 freduce() [1/4]	201
11.6.1.9 freduce() [2/4]	202
11.6.1.10 freduce() [3/4]	202
11.6.1.11 freduce() [4/4]	202
11.6.1.12 fscaln() [1/2]	202
11.6.1.13 fscal() [1/2]	203
11.6.1.14 fscaln() [2/2]	203
11.6.1.15 fscal() [2/2]	203

11.6.1.16 igebb44()	203
11.6.1.17 igebb24()	204
11.6.1.18 igebb14()	204
11.6.1.19 igebb41()	204
11.6.1.20 igebb21()	204
11.6.1.21 igebb11()	205
11.6.1.22 igebp()	205
11.6.1.23 pack_lhs()	205
11.6.1.24 pack_rhs()	206
11.6.1.25 gebp()	206
11.6.1.26 BlockingFactor()	206
11.7 FFLAS::details_spmv Namespace Reference	206
11.8 FFLAS::ElementCategories Namespace Reference	207
11.9 FFLAS::FieldCategories Namespace Reference	207
11.9.1 Detailed Description	207
11.10 FFLAS::MMHelperAlgo Namespace Reference	207
11.11 FFLAS::ModeCategories Namespace Reference	208
11.11.1 Detailed Description	208
11.12 FFLAS::ParSeqHelper Namespace Reference	208
11.12.1 Detailed Description	208
11.13 FFLAS::Protected Namespace Reference	209
11.13.1 Function Documentation	212
11.13.1.1 computeFactorClassic() [1/3]	212
11.13.1.2 computeFactorClassic() [2/3]	212
11.13.1.3 computeFactorClassic() [3/3]	212
11.13.1.4 DotProdBoundClassic()	212
11.13.1.5 TRSMBound() [1/3]	213
11.13.1.6 TRSMBound() [2/3]	213
11.13.1.7 TRSMBound() [3/3]	213
11.13.1.8 fgemm_convert()	213
11.13.1.9 NeedPreAddReduction() [1/2]	214
11.13.1.10 NeedPreAddReduction() [2/2]	214
11.13.1.11 NeedPreSubReduction() [1/2]	214
11.13.1.12 NeedPreSubReduction() [2/2]	214
11.13.1.13 NeedDoublePreAddReduction() [1/2]	215
11.13.1.14 NeedDoublePreAddReduction() [2/2]	215
11.13.1.15 ScalAndReduce() [1/3]	215
11.13.1.16 ScalAndReduce() [2/3]	215
11.13.1.17 fsquareCommon()	216
11.13.1.18 WinogradThreshold() [1/4]	216
11.13.1.19 WinogradThreshold() [2/4]	216
11.13.1.20 WinogradThreshold() [3/4]	216

11.13.1.21 WinogradThreshold() [4/4]	216
11.13.1.22 WinogradSteps()	216
11.13.1.23 DynamicPeeling()	217
11.13.1.24 DynamicPeeling2()	217
11.13.1.25 WinogradCalc()	218
11.13.1.26 fgemv_convert()	218
11.13.1.27 fger_convert()	218
11.13.1.28 fsyrk_convert()	219
11.13.1.29 ScalAndReduce() [3/3]	219
11.13.1.30 NeedPreScalReduction() [1/2]	219
11.13.1.31 NeedPreScalReduction() [2/2]	219
11.13.1.32 NeedPreAxyReduction() [1/2]	220
11.13.1.33 NeedPreAxyReduction() [2/2]	220
11.13.1.34 min_types() [1/7]	220
11.13.1.35 min_types() [2/7]	220
11.13.1.36 min_types() [3/7]	220
11.13.1.37 min_types() [4/7]	220
11.13.1.38 min_types() [5/7]	221
11.13.1.39 min_types() [6/7]	221
11.13.1.40 min_types() [7/7]	221
11.13.1.41 unfit() [1/4]	221
11.13.1.42 unfit() [2/4]	221
11.13.1.43 unfit() [3/4]	221
11.13.1.44 unfit() [4/4]	221
11.13.1.45 igemm_colmajor() [1/2]	222
11.13.1.46 igemm_colmajor() [2/2]	222
11.13.1.47 igemm()	222
11.13.1.48 MatF2MatD_Triangular()	223
11.13.1.49 MatF2MatFI_Triangular()	223
11.14 FFLAS::sell_details Namespace Reference	223
11.15 FFLAS::sparse_details Namespace Reference	223
11.15.1 Function Documentation	226
11.15.1.1 init_y() [1/2]	226
11.15.1.2 init_y() [2/2]	227
11.15.1.3 fspmv_dispatch() [1/2]	227
11.15.1.4 fspmv_dispatch() [2/2]	227
11.15.1.5 fspmv() [1/12]	227
11.15.1.6 fspmv() [2/12]	228
11.15.1.7 fspmv() [3/12]	228
11.15.1.8 fspmv() [4/12]	228
11.15.1.9 fspmv() [5/12]	228
11.15.1.10 fspmv() [6/12]	228

11.15.1.11 fspmv()	[ 7/12]	229
11.15.1.12 fspmv()	[ 8/12]	229
11.15.1.13 fspmv()	[ 9/12]	229
11.15.1.14 fspmm_dispatch()	[ 1/2]	229
11.15.1.15 fspmm_dispatch()	[ 2/2]	230
11.15.1.16 fspmm()	[ 1/9]	230
11.15.1.17 fspmm()	[ 2/9]	230
11.15.1.18 fspmm()	[ 3/9]	230
11.15.1.19 fspmm()	[ 4/9]	231
11.15.1.20 fspmm()	[ 5/9]	231
11.15.1.21 fspmm()	[ 6/9]	231
11.15.1.22 fspmm()	[ 7/9]	231
11.15.1.23 fspmm()	[ 8/9]	232
11.15.1.24 fspmm()	[ 9/9]	232
11.15.1.25 pfspmm_dispatch()	[ 1/2]	232
11.15.1.26 pfspmm_dispatch()	[ 2/2]	232
11.15.1.27 pfspmm()	[ 1/9]	233
11.15.1.28 pfspmm()	[ 2/9]	233
11.15.1.29 pfspmm()	[ 3/9]	233
11.15.1.30 pfspmm()	[ 4/9]	234
11.15.1.31 pfspmm()	[ 5/9]	234
11.15.1.32 pfspmm()	[ 6/9]	234
11.15.1.33 pfspmm()	[ 7/9]	234
11.15.1.34 pfspmm()	[ 8/9]	235
11.15.1.35 pfspmm()	[ 9/9]	235
11.15.1.36 pfspmv()	[ 1/6]	235
11.15.1.37 pfspmv()	[ 2/6]	235
11.15.1.38 pfspmv()	[ 3/6]	236
11.15.1.39 pfspmv()	[ 4/6]	236
11.15.1.40 pfspmv()	[ 5/6]	236
11.15.1.41 pfspmv()	[ 6/6]	236
11.15.1.42 fspmv()	[10/12]	236
11.15.1.43 fspmv()	[11/12]	237
11.15.1.44 fspmv()	[12/12]	237
11.16 FFLAS::sparse_details_impl Namespace Reference		237
11.16.1 Function Documentation		245
11.16.1.1 fspmm()	[ 1/15]	245
11.16.1.2 fspmm()	[ 2/15]	246
11.16.1.3 fspmm()	[ 3/15]	246
11.16.1.4 fspmm_simd_aligned()	[ 1/2]	246
11.16.1.5 fspmm_simd_unaligned()	[ 1/2]	247
11.16.1.6 fspmm_one()	[ 1/4]	247



11.16.1.7 fspmm_mone()	[1/4]	247
11.16.1.8 fspmm_one_simd_aligned()	[1/3]	247
11.16.1.9 fspmm_one_simd_unaligned()	[1/3]	248
11.16.1.10 fspmm_mone_simd_aligned()	[1/3]	248
11.16.1.11 fspmm_mone_simd_unaligned()	[1/3]	248
11.16.1.12 fspmv()	[1/21]	248
11.16.1.13 fspmv()	[2/21]	249
11.16.1.14 fspmv()	[3/21]	249
11.16.1.15 fspmv_one()	[1/10]	249
11.16.1.16 fspmv_mone()	[1/10]	249
11.16.1.17 fspmv_one()	[2/10]	249
11.16.1.18 fspmv_mone()	[2/10]	250
11.16.1.19 pfspmm()	[1/18]	250
11.16.1.20 pfspmm()	[2/18]	250
11.16.1.21 pfspmm()	[3/18]	250
11.16.1.22 pfspmm_one()	[1/2]	251
11.16.1.23 pfspmm_mone()	[1/2]	251
11.16.1.24 pfspmm_one()	[2/2]	251
11.16.1.25 pfspmm_mone()	[2/2]	251
11.16.1.26 pfspmv()	[1/18]	252
11.16.1.27 pfspmv_task()		252
11.16.1.28 pfspmv()	[2/18]	252
11.16.1.29 pfspmv()	[3/18]	252
11.16.1.30 pfspmv_one()	[1/8]	252
11.16.1.31 pfspmv_mone()	[1/8]	253
11.16.1.32 pfspmv_one()	[2/8]	253
11.16.1.33 pfspmv_mone()	[2/8]	253
11.16.1.34 fspmm()	[4/15]	253
11.16.1.35 fspmm()	[5/15]	253
11.16.1.36 fspmm_simd_aligned()	[2/2]	254
11.16.1.37 fspmm_simd_unaligned()	[2/2]	254
11.16.1.38 fspmm()	[6/15]	254
11.16.1.39 fspmm_one()	[2/4]	254
11.16.1.40 fspmm_mone()	[2/4]	255
11.16.1.41 fspmm_one_simd_aligned()	[2/3]	255
11.16.1.42 fspmm_one_simd_unaligned()	[2/3]	255
11.16.1.43 fspmm_mone_simd_aligned()	[2/3]	255
11.16.1.44 fspmm_mone_simd_unaligned()	[2/3]	256
11.16.1.45 fspmv()	[4/21]	256
11.16.1.46 fspmv()	[5/21]	256
11.16.1.47 fspmv()	[6/21]	256
11.16.1.48 fspmv_one()	[3/10]	256

11.16.1.49 fspmv_mone()	[3/10]	257
11.16.1.50 fspmv_one()	[4/10]	257
11.16.1.51 fspmv_mone()	[4/10]	257
11.16.1.52 pfspmm()	[4/18]	257
11.16.1.53 pfspmm()	[5/18]	257
11.16.1.54 pfspmm()	[6/18]	258
11.16.1.55 pfspmm()	[7/18]	258
11.16.1.56 pfspmm()	[8/18]	258
11.16.1.57 pfspmm()	[9/18]	258
11.16.1.58 pfspmv()	[4/18]	259
11.16.1.59 pfspmv()	[5/18]	259
11.16.1.60 pfspmv()	[6/18]	259
11.16.1.61 fspmm()	[7/15]	259
11.16.1.62 fspmm()	[8/15]	259
11.16.1.63 fspmm()	[9/15]	260
11.16.1.64 fspmv()	[7/21]	260
11.16.1.65 fspmv()	[8/21]	260
11.16.1.66 fspmv()	[9/21]	260
11.16.1.67 pfspmm()	[10/18]	260
11.16.1.68 pfspmm()	[11/18]	261
11.16.1.69 pfspmm()	[12/18]	261
11.16.1.70 pfspmm()	[13/18]	261
11.16.1.71 pfspmm()	[14/18]	261
11.16.1.72 pfspmm()	[15/18]	262
11.16.1.73 pfspmm_zo()	[1/2]	262
11.16.1.74 pfspmm_zo()	[2/2]	262
11.16.1.75 pfspmv()	[7/18]	262
11.16.1.76 pfspmv()	[8/18]	262
11.16.1.77 pfspmv()	[9/18]	263
11.16.1.78 pfspmv_one()	[3/8]	263
11.16.1.79 pfspmv_mone()	[3/8]	263
11.16.1.80 pfspmv_one()	[4/8]	263
11.16.1.81 pfspmv_mone()	[4/8]	263
11.16.1.82 fspmm()	[10/15]	264
11.16.1.83 fspmm()	[11/15]	264
11.16.1.84 fspmm()	[12/15]	264
11.16.1.85 fspmm_mone()	[3/4]	264
11.16.1.86 fspmm_one()	[3/4]	265
11.16.1.87 fspmm_mone()	[4/4]	265
11.16.1.88 fspmm_one()	[4/4]	265
11.16.1.89 fspmm_one_simd_aligned()	[3/3]	265
11.16.1.90 fspmm_one_simd_unaligned()	[3/3]	266

11.16.1.91 fspmm_mone_simd_aligned() [3/3]	266
11.16.1.92 fspmm_mone_simd_unaligned() [3/3]	266
11.16.1.93 fspmv() [10/21]	266
11.16.1.94 fspmv() [11/21]	267
11.16.1.95 fspmv() [12/21]	267
11.16.1.96 fspmv_one() [5/10]	267
11.16.1.97 fspmv_mone() [5/10]	267
11.16.1.98 fspmv_one() [6/10]	267
11.16.1.99 fspmv_mone() [6/10]	268
11.16.1.100 pfspmv() [10/18]	268
11.16.1.101 pfspmv() [11/18]	268
11.16.1.102 pfspmv() [12/18]	268
11.16.1.103 pfspmv_one() [5/8]	268
11.16.1.104 pfspmv_mone() [5/8]	269
11.16.1.105 pfspmv_one() [6/8]	269
11.16.1.106 pfspmv_mone() [6/8]	269
11.16.1.107 fspmv() [13/21]	269
11.16.1.108 fspmv_simd() [1/4]	269
11.16.1.109 fspmv() [14/21]	270
11.16.1.110 fspmv_simd() [2/4]	270
11.16.1.111 fspmv() [15/21]	270
11.16.1.112 fspmv_one() [7/10]	270
11.16.1.113 fspmv_mone() [7/10]	270
11.16.1.114 fspmv_one() [8/10]	271
11.16.1.115 fspmv_mone() [8/10]	271
11.16.1.116 fspmv_one_simd() [1/2]	271
11.16.1.117 fspmv_mone_simd() [1/2]	271
11.16.1.118 pfspm() [16/18]	271
11.16.1.119 pfspm() [17/18]	272
11.16.1.120 pfspm() [18/18]	272
11.16.1.121 pfspmv() [13/18]	272
11.16.1.122 pfspmv() [14/18]	272
11.16.1.123 pfspmv() [15/18]	272
11.16.1.124 fspmm() [13/15]	273
11.16.1.125 fspmm() [14/15]	273
11.16.1.126 fspmm() [15/15]	273
11.16.1.127 fspmv() [16/21]	273
11.16.1.128 fspmv() [17/21]	274
11.16.1.129 fspmv() [18/21]	274
11.16.1.130 pfspmv() [16/18]	274
11.16.1.131 pfspmv() [17/18]	274
11.16.1.132 pfspmv() [18/18]	274

11.16.1.133 pfspmv_one() [7/8]	275
11.16.1.134 pfspmv_mone() [7/8]	275
11.16.1.135 pfspmv_one() [8/8]	275
11.16.1.136 pfspmv_mone() [8/8]	275
11.16.1.137 fspmv() [19/21]	275
11.16.1.138 fspmv_simd() [3/4]	276
11.16.1.139 fspmv() [20/21]	276
11.16.1.140 fspmv_simd() [4/4]	276
11.16.1.141 fspmv() [21/21]	276
11.16.1.142 fspmv_one() [9/10]	276
11.16.1.143 fspmv_mone() [9/10]	277
11.16.1.144 fspmv_one_simd() [2/2]	277
11.16.1.145 fspmv_mone_simd() [2/2]	277
11.16.1.146 fspmv_one() [10/10]	277
11.16.1.147 fspmv_mone() [10/10]	277
11.17 FFLAS::StrategyParameter Namespace Reference	278
11.18 FFLAS::StructureHelper Namespace Reference	278
11.18.1 Detailed Description	278
11.19 FFLAS::vectorised Namespace Reference	278
11.19.1 Function Documentation	280
11.19.1.1 VEC_ADD()	280
11.19.1.2 addp()	280
11.19.1.3 VEC_SUB()	280
11.19.1.4 subp()	281
11.19.1.5 add()	281
11.19.1.6 sub()	281
11.19.1.7 axpyp() [1/2]	281
11.19.1.8 axpyp() [2/2]	281
11.19.1.9 reduce() [1/9]	282
11.19.1.10 reduce() [2/9]	282
11.19.1.11 reduce() [3/9]	282
11.19.1.12 reduce() [4/9]	282
11.19.1.13 reduce() [5/9]	282
11.19.1.14 reduce() [6/9]	282
11.19.1.15 reduce() [7/9]	283
11.19.1.16 reduce() [8/9]	283
11.19.1.17 reduce() [9/9]	283
11.19.1.18 modp() [1/2]	283
11.19.1.19 modp() [2/2]	283
11.19.1.20 scalp() [1/3]	283
11.19.1.21 scalp() [2/3]	284
11.19.1.22 scalp() [3/3]	284

11.20 FFLAS::vectorised::unswitch Namespace Reference	284
11.20.1 Function Documentation	285
11.20.1.1 axpyp() [1/2]	285
11.20.1.2 axpyp() [2/2]	285
11.20.1.3 modp() [1/2]	285
11.20.1.4 modp() [2/2]	286
11.20.1.5 scalp() [1/3]	286
11.20.1.6 scalp() [2/3]	286
11.20.1.7 scalp() [3/3]	286
11.21 FFPACK Namespace Reference	287
11.21.1 Detailed Description	303
11.21.2 Typedef Documentation	303
11.21.2.1 Checker_PLUQ	303
11.21.2.2 Checker_Det	303
11.21.2.3 Checker_invert	303
11.21.2.4 Checker_charpoly	303
11.21.2.5 ForceCheck_PLUQ	304
11.21.2.6 ForceCheck_Det	304
11.21.2.7 ForceCheck_invert	304
11.21.2.8 ForceCheck_charpoly	304
11.21.3 Function Documentation	304
11.21.3.1 LAPACKPerm2MathPerm()	304
11.21.3.2 MathPerm2LAPACKPerm()	304
11.21.3.3 applyP() [1/4]	304
11.21.3.4 applyP() [2/4]	305
11.21.3.5 applyP() [3/4]	305
11.21.3.6 MonotonicApplyP()	306
11.21.3.7 fgetrs() [1/4]	307
11.21.3.8 fgetrs() [2/4]	307
11.21.3.9 fgesv() [1/4]	308
11.21.3.10 fgesv() [2/4]	309
11.21.3.11 ftrtri() [1/2]	310
11.21.3.12 trinv_left() [1/2]	310
11.21.3.13 ftrtrm() [1/2]	310
11.21.3.14 ftrstr()	311
11.21.3.15 ftrssyr2k()	311
11.21.3.16 fsytrf() [1/3]	312
11.21.3.17 fsytrf() [2/3]	313
11.21.3.18 fsytrf() [3/3]	313
11.21.3.19 fsytrf_nonunit() [1/3]	313
11.21.3.20 PLUQ() [1/6]	314
11.21.3.21 pPLUQ()	314

11.21.3.22 PLUQ()	[ 2/6 ]	315
11.21.3.23 PLUQ()	[ 3/6 ]	315
11.21.3.24 LUdivine()	[ 1/4 ]	315
11.21.3.25 ColumnEchelonForm()	[ 1/3 ]	316
11.21.3.26 pColumnEchelonForm()		317
11.21.3.27 ColumnEchelonForm()	[ 2/3 ]	317
11.21.3.28 RowEchelonForm()	[ 1/3 ]	317
11.21.3.29 pRowEchelonForm()		318
11.21.3.30 RowEchelonForm()	[ 2/3 ]	318
11.21.3.31 ReducedColumnEchelonForm()	[ 1/3 ]	318
11.21.3.32 pReducedColumnEchelonForm()		319
11.21.3.33 ReducedColumnEchelonForm()	[ 2/3 ]	319
11.21.3.34 ReducedRowEchelonForm()	[ 1/3 ]	320
11.21.3.35 pReducedRowEchelonForm()		320
11.21.3.36 ReducedRowEchelonForm()	[ 2/3 ]	320
11.21.3.37 Invert()	[ 1/4 ]	321
11.21.3.38 Invert()	[ 2/4 ]	321
11.21.3.39 Invert2()	[ 1/2 ]	322
11.21.3.40 CharPoly()	[ 1/8 ]	323
11.21.3.41 CharPoly()	[ 2/8 ]	323
11.21.3.42 CharPoly()	[ 3/8 ]	324
11.21.3.43 MinPoly()	[ 1/4 ]	324
11.21.3.44 MinPoly()	[ 2/4 ]	325
11.21.3.45 MatVecMinPoly()	[ 1/2 ]	325
11.21.3.46 Rank()	[ 1/3 ]	326
11.21.3.47 pRank()		326
11.21.3.48 Rank()	[ 2/3 ]	326
11.21.3.49 IsSingular()	[ 1/2 ]	327
11.21.3.50 Det()	[ 1/6 ]	327
11.21.3.51 pDet()		328
11.21.3.52 Det()	[ 2/6 ]	328
11.21.3.53 Solve()	[ 1/3 ]	328
11.21.3.54 Solve()	[ 2/3 ]	329
11.21.3.55 pSolve()		329
11.21.3.56 RandomNullSpaceVector()	[ 1/3 ]	329
11.21.3.57 NullSpaceBasis()	[ 1/2 ]	330
11.21.3.58 RowRankProfile()	[ 1/3 ]	330
11.21.3.59 pRowRankProfile()		331
11.21.3.60 RowRankProfile()	[ 2/3 ]	331
11.21.3.61 ColumnRankProfile()	[ 1/3 ]	331
11.21.3.62 pColumnRankProfile()		332
11.21.3.63 ColumnRankProfile()	[ 2/3 ]	332

11.21.3.64 RankProfileFromLU()	332
11.21.3.65 LeadingSubmatrixRankProfiles()	333
11.21.3.66 RowRankProfileSubmatrixIndices() [1/2]	333
11.21.3.67 ColRankProfileSubmatrixIndices() [1/2]	334
11.21.3.68 RowRankProfileSubmatrix() [1/2]	335
11.21.3.69 ColRankProfileSubmatrix() [1/2]	335
11.21.3.70 getTriangular() [1/2]	336
11.21.3.71 getTriangular() [2/2]	337
11.21.3.72 getEchelonForm() [1/2]	337
11.21.3.73 getEchelonForm() [2/2]	338
11.21.3.74 getEchelonTransform()	339
11.21.3.75 getReducedEchelonForm() [1/2]	339
11.21.3.76 getReducedEchelonForm() [2/2]	340
11.21.3.77 getReducedEchelonTransform()	341
11.21.3.78 PLUQtoEchelonPermutation()	342
11.21.3.79 LTBruhatGen()	342
11.21.3.80 getLTBruhatGen() [1/2]	342
11.21.3.81 getLTBruhatGen() [2/2]	343
11.21.3.82 LTQSorder()	343
11.21.3.83 CompressToBlockBiDiagonal()	344
11.21.3.84 ExpandBlockBiDiagonalToBruhat()	344
11.21.3.85 Bruhat2EchelonPermutation()	345
11.21.3.86 TInverter() [1/2]	346
11.21.3.87 ComputeRPermutation() [1/2]	346
11.21.3.88 productBruhatxTS() [1/2]	346
11.21.3.89 LQUPtoInverseOfFullRankMinor() [1/2]	347
11.21.3.90 RandomNullSpaceVector() [2/3]	347
11.21.3.91 solveLB() [1/2]	348
11.21.3.92 solveLB2() [1/2]	348
11.21.3.93 TInverter() [2/2]	348
11.21.3.94 ComputeRPermutation() [2/2]	349
11.21.3.95 expandLCRE()	349
11.21.3.96 productBruhatxTS() [2/2]	349
11.21.3.97 Danilevski()	351
11.21.3.98 buildMatrix()	351
11.21.3.99 CharPoly() [4/8]	351
11.21.3.100 CharPoly() [5/8]	351
11.21.3.101 Det() [3/6]	352
11.21.3.102 Det() [4/6]	352
11.21.3.103 fsytrf_BC_Crout()	352
11.21.3.104 fsytrf_BC_RL()	352
11.21.3.105 fsytrf_UP_RPM_BC_RL()	353

11.21.3.106 fsytrf_LOW_RPM_BC_Crout()	353
11.21.3.107 fsytrf_UP_RPM_BC_Crout()	353
11.21.3.108 fsytrf_UP_RPM()	353
11.21.3.109 fsytrf_nonunit() [2/3]	354
11.21.3.110 fsytrf_nonunit() [3/3]	354
11.21.3.111 fsytrf_RPM()	354
11.21.3.112 getTridiagonal()	354
11.21.3.113 LUdivine_gauss() [1/2]	355
11.21.3.114 LUdivine_small() [1/2]	355
11.21.3.115 LUdivine() [2/4]	355
11.21.3.116 LUdivine() [3/4]	356
11.21.3.117 MonotonicCompress()	356
11.21.3.118 MonotonicCompressMorePivots()	356
11.21.3.119 MonotonicCompressCycles()	356
11.21.3.120 MonotonicExpand()	357
11.21.3.121 applyP_block()	357
11.21.3.122 doApplyS()	357
11.21.3.123 MatrixApplyS() [1/3]	358
11.21.3.124 MatrixApplyS() [2/3]	358
11.21.3.125 MatrixApplyS() [3/3]	358
11.21.3.126 PermApplyS()	358
11.21.3.127 doApplyT()	359
11.21.3.128 MatrixApplyT() [1/3]	359
11.21.3.129 MatrixApplyT() [2/3]	359
11.21.3.130 MatrixApplyT() [3/3]	359
11.21.3.131 PermApplyT()	360
11.21.3.132 composePermutationsLLL()	360
11.21.3.133 composePermutationsLLM()	360
11.21.3.134 composePermutationsMLM()	361
11.21.3.135 cyclic_shift_mathPerm()	361
11.21.3.136 cyclic_shift_row_col() [1/2]	361
11.21.3.137 cyclic_shift_row() [1/3]	361
11.21.3.138 cyclic_shift_row() [2/3]	362
11.21.3.139 cyclic_shift_col() [1/3]	362
11.21.3.140 cyclic_shift_col() [2/3]	362
11.21.3.141 PLUQ_basecaseV3()	362
11.21.3.142 PLUQ_basecaseV2()	362
11.21.3.143 PLUQ_basecaseCrout()	363
11.21.3.144 _PLUQ()	363
11.21.3.145 PLUQ() [4/6]	363
11.21.3.146 threads_fgemm()	363
11.21.3.147 threads_ftsm()	364



11.21.3.148 PLUQ() [5/6]	364
11.21.3.149 fflas_const_cast() [1/3]	364
11.21.3.150 fflas_const_cast() [2/3]	364
11.21.3.151 cyclic_shift_row_col() [2/2]	364
11.21.3.152 cyclic_shift_row() [3/3]	365
11.21.3.153 cyclic_shift_col() [3/3]	365
11.21.3.154 applyP() [4/4]	365
11.21.3.155 fgetrs() [3/4]	365
11.21.3.156 fgetrs() [4/4]	366
11.21.3.157 fgesv() [3/4]	366
11.21.3.158 fgesv() [4/4]	366
11.21.3.159 ftrtri() [2/2]	366
11.21.3.160 trinv_left() [2/2]	367
11.21.3.161 ftrtrm() [2/2]	367
11.21.3.162 PLUQ() [6/6]	367
11.21.3.163 LUdivine() [4/4]	367
11.21.3.164 LUdivine_small() [2/2]	368
11.21.3.165 LUdivine_gauss() [2/2]	368
11.21.3.166 RowEchelonForm() [3/3]	368
11.21.3.167 ReducedRowEchelonForm() [3/3]	368
11.21.3.168 ColumnEchelonForm() [3/3]	369
11.21.3.169 ReducedColumnEchelonForm() [3/3]	369
11.21.3.170 Invert() [3/4]	369
11.21.3.171 Invert() [4/4]	369
11.21.3.172 Invert2() [2/2]	369
11.21.3.173 CharPoly() [6/8]	370
11.21.3.174 CharPoly() [7/8]	370
11.21.3.175 CharPoly() [8/8]	370
11.21.3.176 MinPoly() [3/4]	370
11.21.3.177 MinPoly() [4/4]	370
11.21.3.178 MatVecMinPoly() [2/2]	371
11.21.3.179 KrylovElim()	371
11.21.3.180 SpecRankProfile()	371
11.21.3.181 Rank() [3/3]	371
11.21.3.182 IsSingular() [2/2]	371
11.21.3.183 Det() [5/6]	372
11.21.3.184 Det() [6/6]	372
11.21.3.185 Solve() [3/3]	372
11.21.3.186 solveLB() [2/2]	372
11.21.3.187 solveLB2() [2/2]	373
11.21.3.188 RandomNullSpaceVector() [3/3]	373
11.21.3.189 NullSpaceBasis() [2/2]	373

11.21.3.190 RowRankProfile()	[3/3]	373
11.21.3.191 ColumnRankProfile()	[3/3]	374
11.21.3.192 RowRankProfileSubmatrixIndices()	[2/2]	374
11.21.3.193 ColRankProfileSubmatrixIndices()	[2/2]	374
11.21.3.194 RowRankProfileSubmatrix()	[2/2]	374
11.21.3.195 ColRankProfileSubmatrix()	[2/2]	374
11.21.3.196 getTriangular< FFLAS_FIELD< FFLAS_ELT > >()	[1/2]	375
11.21.3.197 getTriangular< FFLAS_FIELD< FFLAS_ELT > >()	[2/2]	375
11.21.3.198 getEchelonForm< FFLAS_FIELD< FFLAS_ELT > >()	[1/2]	375
11.21.3.199 getEchelonForm< FFLAS_FIELD< FFLAS_ELT > >()	[2/2]	375
11.21.3.200 getEchelonTransform< FFLAS_FIELD< FFLAS_ELT > >()		376
11.21.3.201 getReducedEchelonForm< FFLAS_FIELD< FFLAS_ELT > >()	[1/2]	376
11.21.3.202 getReducedEchelonForm< FFLAS_FIELD< FFLAS_ELT > >()	[2/2]	376
11.21.3.203 getReducedEchelonTransform< FFLAS_FIELD< FFLAS_ELT > >()		376
11.21.3.204 LQUPtoInverseOfFullRankMinor()	[2/2]	377
11.21.3.205 fflas_const_cast()	[3/3]	377
11.21.3.206 failure()		377
11.21.3.207 isOdd()	[1/3]	377
11.21.3.208 isOdd()	[2/3]	377
11.21.3.209 isOdd()	[3/3]	377
11.21.3.210 NonZeroRandomMatrix()	[1/2]	378
11.21.3.211 NonZeroRandomMatrix()	[2/2]	378
11.21.3.212 RandomMatrix()	[1/2]	379
11.21.3.213 RandomMatrix()	[2/2]	379
11.21.3.214 RandomTriangularMatrix()	[1/2]	380
11.21.3.215 RandomTriangularMatrix()	[2/2]	380
11.21.3.216 RandInt()		381
11.21.3.217 RandomSymmetricMatrix()		381
11.21.3.218 RandomMatrixWithRank()	[1/2]	382
11.21.3.219 RandomMatrixWithRank()	[2/2]	382
11.21.3.220 RandomIndexSubset()		383
11.21.3.221 RandomPermutation()		383
11.21.3.222 RandomRankProfileMatrix()		383
11.21.3.223 swapval()		384
11.21.3.224 RandomSymmetricRankProfileMatrix()		384
11.21.3.225 RandomLTQSRankProfileMatrix()		384
11.21.3.226 RandomMatrixWithRankandRPM()	[1/2]	384
11.21.3.227 RandomMatrixWithRankandRPM()	[2/2]	385
11.21.3.228 RandomSymmetricMatrixWithRankandRPM()	[1/2]	386
11.21.3.229 RandomSymmetricMatrixWithRankandRPM()	[2/2]	386
11.21.3.230 RandomMatrixWithRankandRandomRPM()	[1/2]	387
11.21.3.231 RandomMatrixWithRankandRandomRPM()	[2/2]	387

11.21.3.232 RandomSymmetricMatrixWithRankandRandomRPM() [1/2]	388
11.21.3.233 RandomSymmetricMatrixWithRankandRandomRPM() [2/2]	388
11.21.3.234 RandomMatrixWithDet() [1/2]	389
11.21.3.235 RandomMatrixWithDet() [2/2]	389
11.21.3.236 RandomLTQSMMatrixWithRankandQSorder()	390
11.21.3.237 chooseField()	390
11.21.3.238 chooseField< Givaro::ZRing< int32_t > >()	390
11.21.3.239 chooseField< Givaro::ZRing< int64_t > >()	390
11.21.3.240 chooseField< Givaro::ZRing< float > >()	391
11.21.3.241 chooseField< Givaro::ZRing< double > >()	391
11.22 FFPACK::Protected Namespace Reference	391
11.22.1 Function Documentation	392
11.22.1.1 LUdivine_construct() [1/2]	392
11.22.1.2 GaussJordan()	393
11.22.1.3 KellerGehrig()	394
11.22.1.4 KGFast()	394
11.22.1.5 KGFast_generalized()	394
11.22.1.6 fgemv_kgf()	394
11.22.1.7 LUKrylov()	394
11.22.1.8 Danilevski()	395
11.22.1.9 RandomKrylovPrecond()	395
11.22.1.10 ArithProg()	395
11.22.1.11 LUKrylov_KGFast()	395
11.22.1.12 MatVecMinPoly()	396
11.22.1.13 Hybrid_KGF_LUK_MinPoly()	396
11.22.1.14 updateD()	396
11.22.1.15 newD()	396
11.22.1.16 CompressRows()	397
11.22.1.17 CompressRowsQK()	397
11.22.1.18 DeCompressRows()	397
11.22.1.19 DeCompressRowsQK()	397
11.22.1.20 CompressRowsQA()	397
11.22.1.21 DeCompressRowsQA()	398
11.22.1.22 LUdivine_construct() [2/2]	398
11.23 Givaro Namespace Reference	398
11.24 MKL_CONFIG Namespace Reference	398
11.25 Reclnt Namespace Reference	398
<b>12 Data Structure Documentation</b>	<b>401</b>
12.1 AlgoChooser< ModeT, ParSeq > Struct Template Reference	401
12.1.1 Member Typedef Documentation	401
12.1.1.1 value	401

12.2 AlgoChooser< ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq > Struct Template Reference	401
12.2.1 Member Typedef Documentation	401
12.2.1.1 value	401
12.3 ALL< v > Struct Template Reference	401
12.4 ALL< false, v... > Struct Template Reference	402
12.4.1 Field Documentation	402
12.4.1.1 value	402
12.5 ALL< true, v... > Struct Template Reference	402
12.5.1 Field Documentation	402
12.5.1.1 value	402
12.6 ALL<> Struct Reference	402
12.6.1 Field Documentation	402
12.6.1.1 value	402
12.7 ArbitraryPrecIntTag Struct Reference	402
12.7.1 Detailed Description	402
12.8 AreEqual< X, Y > Class Template Reference	403
12.8.1 Field Documentation	403
12.8.1.1 value	403
12.9 AreEqual< X, X > Class Template Reference	403
12.9.1 Field Documentation	403
12.9.1.1 value	403
12.10 Argument Struct Reference	403
12.10.1 Field Documentation	403
12.10.1.1 c	403
12.10.1.2 example	403
12.10.1.3 helpString	404
12.10.1.4 type	404
12.10.1.5 data	404
12.11 associatedDelayedField< Field > Struct Template Reference	404
12.11.1 Member Typedef Documentation	404
12.11.1.1 field	404
12.11.1.2 type	404
12.12 associatedDelayedField< const FFPACK::RNSIntegerMod< RNS > > Struct Template Reference	404
12.12.1 Member Typedef Documentation	404
12.12.1.1 field	404
12.12.1.2 type	405
12.13 associatedDelayedField< const Givaro::Modular< T, X > > Struct Template Reference	405
12.13.1 Member Typedef Documentation	405
12.13.1.1 field	405
12.13.1.2 type	405
12.14 associatedDelayedField< const Givaro::ModularBalanced< T > > Struct Template Reference	405

12.14.1 Member Typedef Documentation	405
12.14.1.1 field	405
12.14.1.2 type	405
12.15 associatedDelayedField< const Givaro::ZRing< T > > Struct Template Reference	406
12.15.1 Member Typedef Documentation	406
12.15.1.1 field	406
12.15.1.2 type	406
12.16 Auto Struct Reference	406
12.17 Bench< Elt > Class Template Reference	406
12.17.1 Member Typedef Documentation	407
12.17.1.1 Field	407
12.17.1.2 Elt_ptr	407
12.17.1.3 Residu	407
12.17.1.4 enable_if_t	407
12.17.1.5 is_same_element	407
12.17.1.6 enable_if_no_simd_t	407
12.17.1.7 enable_if_simd128_t	407
12.17.1.8 enable_if_simd256_t	407
12.17.1.9 enable_if_simd512_t	408
12.17.2 Constructor & Destructor Documentation	408
12.17.2.1 Bench()	408
12.17.3 Member Function Documentation	408
12.17.3.1 cardinality() [1/2]	408
12.17.3.2 cardinality() [2/2]	408
12.17.3.3 doBenches()	408
12.17.3.4 run()	408
12.17.4 Field Documentation	408
12.17.4.1 F	408
12.17.4.2 m	408
12.17.4.3 n	409
12.17.4.4 iters	409
12.17.4.5 inplace	409
12.18 Bini Struct Reference	409
12.19 Block Struct Reference	409
12.20 BlockTransposeSIMD< Field, Simd, > Struct Template Reference	409
12.20.1 Member Function Documentation	409
12.20.1.1 size()	409
12.20.1.2 info()	410
12.20.1.3 transpose() [1/5]	410
12.20.1.4 transpose() [2/5]	410
12.20.1.5 transpose() [3/5]	410
12.20.1.6 transpose() [4/5]	410

12.20.1.7 transpose() [5/5]	410
12.21 callLUdivine_small< Element > Class Template Reference	411
12.21.1 Member Function Documentation	411
12.21.1.1 operator()()	411
12.22 callLUdivine_small< double > Class Reference	411
12.22.1 Member Function Documentation	411
12.22.1.1 operator()()	411
12.23 callLUdivine_small< float > Class Reference	412
12.23.1 Member Function Documentation	412
12.23.1.1 operator()()	412
12.24 CharpolyFailed Class Reference	412
12.25 Checker_Empty< Field > Struct Template Reference	412
12.25.1 Constructor & Destructor Documentation	413
12.25.1.1 Checker_Empty()	413
12.25.2 Member Function Documentation	413
12.25.2.1 check()	413
12.26 CheckerImplem_charpoly< Field, Polynomial > Class Template Reference	413
12.26.1 Constructor & Destructor Documentation	413
12.26.1.1 CheckerImplem_charpoly() [1/2]	413
12.26.1.2 CheckerImplem_charpoly() [2/2]	413
12.26.1.3 ~CheckerImplem_charpoly()	413
12.26.2 Member Function Documentation	414
12.26.2.1 check()	414
12.27 CheckerImplem_charpoly< Givaro::ZRing< Givaro::Integer >, Polynomial > Class Template Reference	414
12.27.1 Member Typedef Documentation	414
12.27.1.1 Ring	414
12.27.2 Constructor & Destructor Documentation	414
12.27.2.1 CheckerImplem_charpoly() [1/2]	414
12.27.2.2 CheckerImplem_charpoly() [2/2]	414
12.27.2.3 ~CheckerImplem_charpoly()	414
12.27.3 Member Function Documentation	415
12.27.3.1 check()	415
12.28 CheckerImplem_Det< Field > Class Template Reference	415
12.28.1 Constructor & Destructor Documentation	415
12.28.1.1 CheckerImplem_Det() [1/2]	415
12.28.1.2 CheckerImplem_Det() [2/2]	415
12.28.1.3 ~CheckerImplem_Det()	415
12.28.2 Member Function Documentation	415
12.28.2.1 check()	415
12.29 CheckerImplem_fgemm< Field > Class Template Reference	417
12.29.1 Constructor & Destructor Documentation	417

12.29.1.1 CheckerImplem_fgemm() [1/2]	417
12.29.1.2 CheckerImplem_fgemm() [2/2]	417
12.29.1.3 ~CheckerImplem_fgemm()	417
12.29.2 Member Function Documentation	418
12.29.2.1 check()	418
12.30 CheckerImplem_ftsm< Field > Class Template Reference	418
12.30.1 Constructor & Destructor Documentation	418
12.30.1.1 CheckerImplem_ftsm() [1/2]	418
12.30.1.2 CheckerImplem_ftsm() [2/2]	418
12.30.1.3 ~CheckerImplem_ftsm()	418
12.30.2 Member Function Documentation	419
12.30.2.1 check()	419
12.31 CheckerImplem_invert< Field > Class Template Reference	419
12.31.1 Constructor & Destructor Documentation	419
12.31.1.1 CheckerImplem_invert() [1/2]	419
12.31.1.2 CheckerImplem_invert() [2/2]	419
12.31.1.3 ~CheckerImplem_invert()	419
12.31.2 Member Function Documentation	420
12.31.2.1 check()	420
12.32 CheckerImplem_PLUQ< Field > Class Template Reference	420
12.32.1 Constructor & Destructor Documentation	420
12.32.1.1 CheckerImplem_PLUQ() [1/2]	420
12.32.1.2 CheckerImplem_PLUQ() [2/2]	420
12.32.1.3 ~CheckerImplem_PLUQ()	420
12.32.2 Member Function Documentation	420
12.32.2.1 check()	420
12.33 Classic Struct Reference	421
12.34 Column Struct Reference	421
12.35 CompactElement< Element > Struct Template Reference	421
12.35.1 Member Typedef Documentation	421
12.35.1.1 type	421
12.36 CompactElement< double > Struct Reference	421
12.36.1 Member Typedef Documentation	422
12.36.1.1 type	422
12.37 CompactElement< float > Struct Reference	422
12.37.1 Member Typedef Documentation	422
12.37.1.1 type	422
12.38 CompactElement< int16_t > Struct Reference	422
12.38.1 Member Typedef Documentation	422
12.38.1.1 type	422
12.39 CompactElement< int32_t > Struct Reference	422
12.39.1 Member Typedef Documentation	422

12.39.1.1 type	422
12.40 CompactElement< int64_t > Struct Reference	422
12.40.1 Member Typedef Documentation	423
12.40.1.1 type	423
12.41 compatible_data_type< Field > Struct Template Reference	423
12.41.1 Field Documentation	423
12.41.1.1 value	423
12.42 compatible_data_type< Givaro::ZRing< double > > Struct Reference	423
12.42.1 Field Documentation	423
12.42.1.1 value	423
12.43 compatible_data_type< Givaro::ZRing< float > > Struct Reference	423
12.43.1 Field Documentation	423
12.43.1.1 value	423
12.44 Compose< H1, H2 > Struct Template Reference	424
12.44.1 Constructor & Destructor Documentation	424
12.44.1.1 Compose() [1/5]	424
12.44.1.2 Compose() [2/5]	424
12.44.1.3 Compose() [3/5]	424
12.44.1.4 Compose() [4/5]	424
12.44.1.5 Compose() [5/5]	424
12.44.2 Member Function Documentation	424
12.44.2.1 first_component()	424
12.44.2.2 second_component()	424
12.44.3 Friends And Related Symbol Documentation	425
12.44.3.1 operator<<	425
12.45 Simd128_impl< true, true, false, 2 >::Converter Union Reference	425
12.45.1 Field Documentation	425
12.45.1.1 v	425
12.45.1.2 t	425
12.46 Simd128_impl< true, true, false, 4 >::Converter Union Reference	425
12.46.1 Field Documentation	425
12.46.1.1 v	425
12.46.1.2 t	425
12.47 Simd128_impl< true, true, false, 8 >::Converter Union Reference	425
12.47.1 Field Documentation	426
12.47.1.1 v	426
12.47.1.2 t	426
12.48 Simd128_impl< true, true, true, 2 >::Converter Union Reference	426
12.48.1 Field Documentation	426
12.48.1.1 v	426
12.48.1.2 t	426
12.49 Simd128_impl< true, true, true, 4 >::Converter Union Reference	426



12.49.1 Field Documentation	426
12.49.1.1 v	426
12.49.1.2 t	426
12.50 Simd128_impl< true, true, true, 8 >::Converter Union Reference	426
12.50.1 Field Documentation	427
12.50.1.1 v	427
12.50.1.2 t	427
12.51 Simd256_impl< true, false, true, 8 >::Converter Union Reference	427
12.51.1 Field Documentation	427
12.51.1.1 v	427
12.51.1.2 t	427
12.52 Simd256_impl< true, true, false, 2 >::Converter Union Reference	427
12.52.1 Field Documentation	427
12.52.1.1 v	427
12.52.1.2 t	427
12.53 Simd256_impl< true, true, false, 4 >::Converter Union Reference	427
12.53.1 Field Documentation	428
12.53.1.1 v	428
12.53.1.2 t	428
12.54 Simd256_impl< true, true, false, 8 >::Converter Union Reference	428
12.54.1 Field Documentation	428
12.54.1.1 v	428
12.54.1.2 t	428
12.55 Simd256_impl< true, true, true, 2 >::Converter Union Reference	428
12.55.1 Field Documentation	428
12.55.1.1 v	428
12.55.1.2 t	428
12.56 Simd256_impl< true, true, true, 4 >::Converter Union Reference	428
12.56.1 Field Documentation	429
12.56.1.1 v	429
12.56.1.2 t	429
12.57 Simd256_impl< true, true, true, 8 >::Converter Union Reference	429
12.57.1 Field Documentation	429
12.57.1.1 v	429
12.57.1.2 t	429
12.58 Simd512_impl< true, true, false, 8 >::Converter Union Reference	429
12.58.1 Field Documentation	429
12.58.1.1 v	429
12.58.1.2 t	429
12.59 Simd512_impl< true, true, true, 8 >::Converter Union Reference	429
12.59.1 Field Documentation	430
12.59.1.1 v	430

12.59.1.2 t . . . . .	430
12.60 ConvertTo< T > Struct Template Reference . . . . .	430
12.60.1 Detailed Description . . . . .	430
12.61 Coo< ValT, IdxT > Struct Template Reference . . . . .	430
12.61.1 Member Typedef Documentation . . . . .	431
12.61.1.1 Self . . . . .	431
12.61.2 Constructor & Destructor Documentation . . . . .	431
12.61.2.1 Coo() [1/4] . . . . .	431
12.61.2.2 Coo() [2/4] . . . . .	431
12.61.2.3 Coo() [3/4] . . . . .	431
12.61.2.4 Coo() [4/4] . . . . .	431
12.61.3 Member Function Documentation . . . . .	431
12.61.3.1 operator=() [1/2] . . . . .	431
12.61.3.2 operator=() [2/2] . . . . .	431
12.61.4 Field Documentation . . . . .	431
12.61.4.1 val . . . . .	431
12.61.4.2 row . . . . .	431
12.61.4.3 col . . . . .	432
12.62 Coo< Field > Struct Template Reference . . . . .	432
12.62.1 Constructor & Destructor Documentation . . . . .	432
12.62.1.1 Coo() [1/4] . . . . .	432
12.62.1.2 Coo() [2/4] . . . . .	432
12.62.1.3 Coo() [3/4] . . . . .	432
12.62.1.4 Coo() [4/4] . . . . .	432
12.62.2 Member Function Documentation . . . . .	433
12.62.2.1 operator=() [1/2] . . . . .	433
12.62.2.2 operator=() [2/2] . . . . .	433
12.62.3 Field Documentation . . . . .	433
12.62.3.1 val . . . . .	433
12.62.3.2 col . . . . .	433
12.62.3.3 row . . . . .	433
12.62.3.4 deleted . . . . .	433
12.63 Coo< ValT, IdxT > Struct Template Reference . . . . .	433
12.63.1 Member Typedef Documentation . . . . .	434
12.63.1.1 Self . . . . .	434
12.63.2 Constructor & Destructor Documentation . . . . .	434
12.63.2.1 Coo() [1/4] . . . . .	434
12.63.2.2 Coo() [2/4] . . . . .	434
12.63.2.3 Coo() [3/4] . . . . .	434
12.63.2.4 Coo() [4/4] . . . . .	434
12.63.3 Member Function Documentation . . . . .	434
12.63.3.1 operator=() [1/2] . . . . .	434

12.63.3.2 operator=() [2/2]	434
12.63.4 Field Documentation	434
12.63.4.1 val	434
12.63.4.2 row	434
12.63.4.3 col	435
12.64 CooMat< Field > Struct Template Reference	435
12.64.1 Field Documentation	435
12.64.1.1 _coo16	435
12.64.1.2 _coo32	435
12.64.1.3 _coo64	435
12.64.1.4 _coo16_zo	435
12.64.1.5 _coo32_zo	435
12.64.1.6 _coo64_zo	435
12.65 count_nonconst_lvalue_reference< T > Struct Template Reference	436
12.66 count_nonconst_lvalue_reference< const T &, O... > Struct Template Reference	436
12.66.1 Field Documentation	436
12.66.1.1 n	436
12.67 count_nonconst_lvalue_reference< T &, O... > Struct Template Reference	436
12.67.1 Field Documentation	436
12.67.1.1 n	436
12.68 count_nonconst_lvalue_reference< T, O... > Struct Template Reference	436
12.68.1 Field Documentation	436
12.68.1.1 n	436
12.69 count_nonconst_lvalue_reference<> Struct Reference	437
12.69.1 Field Documentation	437
12.69.1.1 n	437
12.70 CsrMat< Field > Struct Template Reference	437
12.70.1 Field Documentation	437
12.70.1.1 _csr16	437
12.70.1.2 _csr32	437
12.70.1.3 _csr64	437
12.70.1.4 _csr16_zo	437
12.70.1.5 _csr32_zo	437
12.70.1.6 _csr64_zo	437
12.71 DefaultBoundedTag Struct Reference	438
12.71.1 Detailed Description	438
12.72 DefaultTag Struct Reference	438
12.72.1 Detailed Description	438
12.73 DelayedTag Struct Reference	438
12.73.1 Detailed Description	438
12.74 DivideAndConquer Struct Reference	438
12.75 ElementTraits< Element > Struct Template Reference	438

12.75.1 Detailed Description . . . . .	438
12.75.2 Member Typedef Documentation . . . . .	439
12.75.2.1 value . . . . .	439
12.76 ElementTraits< double > Struct Reference . . . . .	439
12.76.1 Member Typedef Documentation . . . . .	439
12.76.1.1 value . . . . .	439
12.77 ElementTraits< FFPACK::rns_double_elt > Struct Reference . . . . .	439
12.77.1 Member Typedef Documentation . . . . .	439
12.77.1.1 value . . . . .	439
12.78 ElementTraits< float > Struct Reference . . . . .	439
12.78.1 Member Typedef Documentation . . . . .	439
12.78.1.1 value . . . . .	439
12.79 ElementTraits< Givaro::Integer > Struct Reference . . . . .	440
12.79.1 Member Typedef Documentation . . . . .	440
12.79.1.1 value . . . . .	440
12.80 ElementTraits< int16_t > Struct Reference . . . . .	440
12.80.1 Member Typedef Documentation . . . . .	440
12.80.1.1 value . . . . .	440
12.81 ElementTraits< int32_t > Struct Reference . . . . .	440
12.81.1 Member Typedef Documentation . . . . .	440
12.81.1.1 value . . . . .	440
12.82 ElementTraits< int64_t > Struct Reference . . . . .	440
12.82.1 Member Typedef Documentation . . . . .	441
12.82.1.1 value . . . . .	441
12.83 ElementTraits< int8_t > Struct Reference . . . . .	441
12.83.1 Member Typedef Documentation . . . . .	441
12.83.1.1 value . . . . .	441
12.84 ElementTraits< RecInt::rint< K > > Struct Template Reference . . . . .	441
12.84.1 Member Typedef Documentation . . . . .	441
12.84.1.1 value . . . . .	441
12.85 ElementTraits< RecInt::rmint< K, MG > > Struct Template Reference . . . . .	441
12.85.1 Member Typedef Documentation . . . . .	441
12.85.1.1 value . . . . .	441
12.86 ElementTraits< RecInt::ruint< K > > Struct Template Reference . . . . .	442
12.86.1 Member Typedef Documentation . . . . .	442
12.86.1.1 value . . . . .	442
12.87 ElementTraits< uint16_t > Struct Reference . . . . .	442
12.87.1 Member Typedef Documentation . . . . .	442
12.87.1.1 value . . . . .	442
12.88 ElementTraits< uint32_t > Struct Reference . . . . .	442
12.88.1 Member Typedef Documentation . . . . .	442
12.88.1.1 value . . . . .	442

12.89 ElementTraits< uint64_t > Struct Reference . . . . .	442
12.89.1 Member Typedef Documentation . . . . .	443
12.89.1.1 value . . . . .	443
12.90 ElementTraits< uint8_t > Struct Reference . . . . .	443
12.90.1 Member Typedef Documentation . . . . .	443
12.90.1.1 value . . . . .	443
12.91 EllMat< Field > Struct Template Reference . . . . .	443
12.91.1 Field Documentation . . . . .	443
12.91.1.1 _ell16 . . . . .	443
12.91.1.2 _ell32 . . . . .	443
12.91.1.3 _ell64 . . . . .	443
12.91.1.4 _ell16_zo . . . . .	444
12.91.1.5 _ell32_zo . . . . .	444
12.91.1.6 _ell64_zo . . . . .	444
12.92 Failure Class Reference . . . . .	444
12.92.1 Detailed Description . . . . .	444
12.92.2 Constructor & Destructor Documentation . . . . .	444
12.92.2.1 Failure() . . . . .	444
12.92.3 Member Function Documentation . . . . .	444
12.92.3.1 operator>() [1/2] . . . . .	444
12.92.3.2 operator>() [2/2] . . . . .	445
12.92.3.3 setErrorStream() . . . . .	445
12.92.3.4 print() . . . . .	445
12.92.4 Field Documentation . . . . .	445
12.92.4.1 _errorStream . . . . .	445
12.93 FailureCharpolyCheck Class Reference . . . . .	445
12.94 FailureDetCheck Class Reference . . . . .	446
12.95 FailureFgemmCheck Class Reference . . . . .	446
12.96 FailureInvertCheck Class Reference . . . . .	446
12.97 FailurePLUQCheck Class Reference . . . . .	446
12.98 FailureTrsmCheck Class Reference . . . . .	446
12.99 FieldSimd< _Field > Class Template Reference . . . . .	446
12.99.1 Member Typedef Documentation . . . . .	447
12.99.1.1 Field . . . . .	447
12.99.1.2 Element . . . . .	447
12.99.1.3 simd . . . . .	447
12.99.1.4 vect_t . . . . .	447
12.99.1.5 scalar_t . . . . .	448
12.99.2 Constructor & Destructor Documentation . . . . .	448
12.99.2.1 FieldSimd() [1/3] . . . . .	448
12.99.2.2 FieldSimd() [2/3] . . . . .	448
12.99.2.3 FieldSimd() [3/3] . . . . .	448

12.99.3 Member Function Documentation	448
12.99.3.1 operator=() [1/2]	448
12.99.3.2 operator=() [2/2]	448
12.99.3.3 init() [1/2]	448
12.99.3.4 init() [2/2]	448
12.99.3.5 add() [1/2]	448
12.99.3.6 add() [2/2]	449
12.99.3.7 addin()	449
12.99.3.8 add_r() [1/2]	449
12.99.3.9 add_r() [2/2]	449
12.99.3.10 addin_r()	449
12.99.3.11 sub() [1/2]	449
12.99.3.12 sub() [2/2]	449
12.99.3.13 subin()	449
12.99.3.14 sub_r() [1/2]	450
12.99.3.15 sub_r() [2/2]	450
12.99.3.16 subin_r()	450
12.99.3.17 zero() [1/2]	450
12.99.3.18 zero() [2/2]	450
12.99.3.19 mod()	450
12.99.3.20 mul() [1/2]	450
12.99.3.21 mul() [2/2]	450
12.99.3.22 mulin()	450
12.99.3.23 mul_r() [1/2]	451
12.99.3.24 mul_r() [2/2]	451
12.99.3.25 axpy() [1/2]	451
12.99.3.26 axpy() [2/2]	451
12.99.3.27 axpyin()	451
12.99.3.28 axpy_r() [1/2]	451
12.99.3.29 axpy_r() [2/2]	451
12.99.3.30 axpyin_r()	452
12.99.3.31 maxpy() [1/2]	452
12.99.3.32 maxpy() [2/2]	452
12.99.3.33 maxpyin()	452
12.99.4 Field Documentation	452
12.99.4.1 vect_size	452
12.99.4.2 alignment	452
12.100 FieldTraits< Field > Struct Template Reference	452
12.100.1 Detailed Description	453
12.100.2 Member Typedef Documentation	453
12.100.2.1 category	453
12.100.3 Field Documentation	453

12.100.3.1 balanced	453
12.101 FieldTraits< FFPACK::RNSInteger< T > > Struct Template Reference	453
12.101.1 Member Typedef Documentation	453
12.101.1.1 category	453
12.101.2 Field Documentation	453
12.101.2.1 balanced	453
12.102 FieldTraits< FFPACK::RNSIntegerMod< T > > Struct Template Reference	453
12.102.1 Member Typedef Documentation	454
12.102.1.1 category	454
12.102.2 Field Documentation	454
12.102.2.1 balanced	454
12.103 FieldTraits< Givaro::Modular< Element > > Struct Template Reference	454
12.103.1 Member Typedef Documentation	454
12.103.1.1 category	454
12.103.2 Field Documentation	454
12.103.2.1 balanced	454
12.104 FieldTraits< Givaro::ModularBalanced< Element > > Struct Template Reference	454
12.104.1 Member Typedef Documentation	455
12.104.1.1 category	455
12.104.2 Field Documentation	455
12.104.2.1 balanced	455
12.105 FieldTraits< Givaro::ZRing< double > > Struct Reference	455
12.105.1 Member Typedef Documentation	455
12.105.1.1 category	455
12.105.2 Field Documentation	455
12.105.2.1 balanced	455
12.106 FieldTraits< Givaro::ZRing< float > > Struct Reference	455
12.106.1 Member Typedef Documentation	456
12.106.1.1 category	456
12.106.2 Field Documentation	456
12.106.2.1 balanced	456
12.107 FieldTraits< Givaro::ZRing< Givaro::Integer > > Struct Reference	456
12.107.1 Member Typedef Documentation	456
12.107.1.1 category	456
12.107.2 Field Documentation	456
12.107.2.1 balanced	456
12.108 FieldTraits< Givaro::ZRing< int16_t > > Struct Reference	456
12.108.1 Member Typedef Documentation	457
12.108.1.1 category	457
12.108.2 Field Documentation	457
12.108.2.1 balanced	457
12.109 FieldTraits< Givaro::ZRing< int32_t > > Struct Reference	457

12.109.1 Member Typedef Documentation	457
12.109.1.1 category	457
12.109.2 Field Documentation	457
12.109.2.1 balanced	457
12.110 FieldTraits< Givaro::ZRing< int64_t > > Struct Reference	457
12.110.1 Member Typedef Documentation	457
12.110.1.1 category	457
12.110.2 Field Documentation	458
12.110.2.1 balanced	458
12.111 FieldTraits< Givaro::ZRing< Reclnt::ruint< K > > > Struct Template Reference	458
12.111.1 Member Typedef Documentation	458
12.111.1.1 category	458
12.111.2 Field Documentation	458
12.111.2.1 balanced	458
12.112 FieldTraits< Givaro::ZRing< uint16_t > > Struct Reference	458
12.112.1 Member Typedef Documentation	458
12.112.1.1 category	458
12.112.2 Field Documentation	459
12.112.2.1 balanced	459
12.113 FieldTraits< Givaro::ZRing< uint32_t > > Struct Reference	459
12.113.1 Member Typedef Documentation	459
12.113.1.1 category	459
12.113.2 Field Documentation	459
12.113.2.1 balanced	459
12.114 FieldTraits< Givaro::ZRing< uint64_t > > Struct Reference	459
12.114.1 Member Typedef Documentation	459
12.114.1.1 category	459
12.114.2 Field Documentation	459
12.114.2.1 balanced	459
12.115 Fixed Struct Reference	460
12.116 FixedPreclntTag Struct Reference	460
12.116.1 Detailed Description	460
12.117 ScalFunctionsBase< Element, typename enable_if< is_floating_point< Element >::value >::type >::FloatingPointTestDistribution Class Reference	460
12.117.1 Member Typedef Documentation	460
12.117.1.1 IntType	460
12.117.2 Constructor & Destructor Documentation	460
12.117.2.1 FloatingPointTestDistribution()	460
12.117.3 Member Function Documentation	460
12.117.3.1 operator>()	460
12.118 ForStrategy1D< blocksize_t, Cut, Param > Struct Template Reference	461
12.118.1 Constructor & Destructor Documentation	461



12.118.1.1 ForStrategy1D() [1/2]	461
12.118.1.2 ForStrategy1D() [2/2]	461
12.118.2 Member Function Documentation	461
12.118.2.1 build()	461
12.118.2.2 initialize()	461
12.118.2.3 isTerminated()	462
12.118.2.4 begin()	462
12.118.2.5 end()	462
12.118.2.6 numblocks()	462
12.118.2.7 blockindex()	462
12.118.2.8 operator++()	462
12.118.3 Field Documentation	462
12.118.3.1 ibeg	462
12.118.3.2 iend	462
12.118.3.3 current	462
12.118.3.4 firstBlockSize	462
12.118.3.5 lastBlockSize	463
12.118.3.6 changeBS	463
12.118.3.7 numBlock	463
12.119 ForStrategy2D< blocksize_t, Cut, Param > Struct Template Reference	463
12.119.1 Constructor & Destructor Documentation	464
12.119.1.1 ForStrategy2D()	464
12.119.2 Member Function Documentation	464
12.119.2.1 initialize()	464
12.119.2.2 isTerminated()	464
12.119.2.3 ibegin()	464
12.119.2.4 jbegin()	464
12.119.2.5 iend()	464
12.119.2.6 jend()	464
12.119.2.7 operator++()	464
12.119.2.8 rownumblocks()	465
12.119.2.9 colnumblocks()	465
12.119.2.10 blockindex()	465
12.119.2.11 rowblockindex()	465
12.119.2.12 colblockindex()	465
12.119.3 Friends And Related Symbol Documentation	465
12.119.3.1 operator<<	465
12.119.4 Field Documentation	465
12.119.4.1 _ibeg	465
12.119.4.2 _iend	465
12.119.4.3 _jbeg	465
12.119.4.4 _jend	466

12.119.4.5 rowBlockSize . . . . .	466
12.119.4.6 colBlockSize . . . . .	466
12.119.4.7 current . . . . .	466
12.119.4.8 lastRBS . . . . .	466
12.119.4.9 lastCBS . . . . .	466
12.119.4.10 changeRBS . . . . .	466
12.119.4.11 changeCBS . . . . .	466
12.119.4.12 numRowsBlock . . . . .	466
12.119.4.13 numColBlock . . . . .	466
12.119.4.14 BLOCKS . . . . .	467
12.120 ftrmmLeftLowerNoTransNonUnit< Element > Class Template Reference . . . . .	467
12.121 ftrmmLeftLowerNoTransUnit< Element > Class Template Reference . . . . .	467
12.122 ftrmmLeftLowerTransNonUnit< Element > Class Template Reference . . . . .	467
12.123 ftrmmLeftLowerTransUnit< Element > Class Template Reference . . . . .	467
12.124 ftrmmLeftUpperNoTransNonUnit< Element > Class Template Reference . . . . .	467
12.125 ftrmmLeftUpperNoTransUnit< Element > Class Template Reference . . . . .	467
12.126 ftrmmLeftUpperTransNonUnit< Element > Class Template Reference . . . . .	467
12.127 ftrmmLeftUpperTransUnit< Element > Class Template Reference . . . . .	468
12.128 ftrmmRightLowerNoTransNonUnit< Element > Class Template Reference . . . . .	468
12.129 ftrmmRightLowerNoTransUnit< Element > Class Template Reference . . . . .	468
12.130 ftrmmRightLowerTransNonUnit< Element > Class Template Reference . . . . .	468
12.131 ftrmmRightLowerTransUnit< Element > Class Template Reference . . . . .	468
12.132 ftrmmRightUpperNoTransNonUnit< Element > Class Template Reference . . . . .	468
12.133 ftrmmRightUpperNoTransUnit< Element > Class Template Reference . . . . .	468
12.134 ftrmmRightUpperTransNonUnit< Element > Class Template Reference . . . . .	468
12.135 ftrmmRightUpperTransUnit< Element > Class Template Reference . . . . .	469
12.136 frsmLeftLowerNoTransNonUnit< Element > Class Template Reference . . . . .	469
12.137 frsmLeftLowerNoTransUnit< Element > Class Template Reference . . . . .	469
12.138 frsmLeftLowerTransNonUnit< Element > Class Template Reference . . . . .	469
12.139 frsmLeftLowerTransUnit< Element > Class Template Reference . . . . .	469
12.140 frsmLeftUpperNoTransNonUnit< Element > Class Template Reference . . . . .	469
12.140.1 Detailed Description . . . . .	469
12.141 frsmLeftUpperNoTransUnit< Element > Class Template Reference . . . . .	470
12.142 frsmLeftUpperTransNonUnit< Element > Class Template Reference . . . . .	470
12.143 frsmLeftUpperTransUnit< Element > Class Template Reference . . . . .	470
12.144 frsmRightLowerNoTransNonUnit< Element > Class Template Reference . . . . .	470
12.145 frsmRightLowerNoTransUnit< Element > Class Template Reference . . . . .	470
12.146 frsmRightLowerTransNonUnit< Element > Class Template Reference . . . . .	470
12.147 frsmRightLowerTransUnit< Element > Class Template Reference . . . . .	470
12.148 frsmRightUpperNoTransNonUnit< Element > Class Template Reference . . . . .	470
12.149 frsmRightUpperNoTransUnit< Element > Class Template Reference . . . . .	471
12.150 frsmRightUpperTransNonUnit< Element > Class Template Reference . . . . .	471

12.151 frsmRightUpperTransUnit< Element > Class Template Reference . . . . .	471
12.152 GenericTag Struct Reference . . . . .	471
12.152.1 Detailed Description . . . . .	471
12.153 GenericTag Struct Reference . . . . .	471
12.153.1 Detailed Description . . . . .	471
12.154 Grain Struct Reference . . . . .	471
12.155 has_minus_eq_impl< C > Struct Template Reference . . . . .	471
12.155.1 Field Documentation . . . . .	472
12.155.1.1 value . . . . .	472
12.156 has_minus_impl< C > Struct Template Reference . . . . .	472
12.156.1 Field Documentation . . . . .	472
12.156.1.1 value . . . . .	472
12.157 has_mul_eq_impl< C > Struct Template Reference . . . . .	472
12.157.1 Field Documentation . . . . .	472
12.157.1.1 value . . . . .	472
12.158 has_mul_impl< C > Struct Template Reference . . . . .	472
12.158.1 Field Documentation . . . . .	473
12.158.1.1 value . . . . .	473
12.159 has_operation< T > Struct Template Reference . . . . .	473
12.159.1 Field Documentation . . . . .	473
12.159.1.1 value . . . . .	473
12.160 has_plus_eq_impl< C > Struct Template Reference . . . . .	473
12.160.1 Field Documentation . . . . .	473
12.160.1.1 value . . . . .	473
12.161 has_plus_impl< C > Struct Template Reference . . . . .	473
12.161.1 Field Documentation . . . . .	474
12.161.1.1 value . . . . .	474
12.162 HelperFlag Struct Reference . . . . .	474
12.162.1 Field Documentation . . . . .	474
12.162.1.1 none . . . . .	474
12.162.1.2 coo . . . . .	474
12.162.1.3 csr . . . . .	474
12.162.1.4 ell . . . . .	474
12.162.1.5 aut . . . . .	474
12.162.1.6 pm1 . . . . .	474
12.163 HelperMod< Field, ElementTraits > Struct Template Reference . . . . .	474
12.164 HelperMod< Field, ElementCategories::MachineIntTag > Struct Template Reference . . . . .	475
12.164.1 Constructor & Destructor Documentation . . . . .	475
12.164.1.1 HelperMod() [1/2] . . . . .	475
12.164.1.2 HelperMod() [2/2] . . . . .	475
12.164.2 Field Documentation . . . . .	475
12.164.2.1 p . . . . .	475

12.164.2.2 invp	475
12.164.2.3 min	475
12.164.2.4 max	475
12.164.2.5 pow50rem	475
12.165 HelperMod< Field, FFLAS::ElementCategories::ArbitraryPrecIntTag > Struct Template Reference	476
12.165.1 Constructor & Destructor Documentation	476
12.165.1.1 HelperMod() [1/2]	476
12.165.1.2 HelperMod() [2/2]	476
12.165.2 Field Documentation	476
12.165.2.1 p	476
12.166 HelperMod< Field, FFLAS::ElementCategories::FixedPrecIntTag > Struct Template Reference	476
12.166.1 Constructor & Destructor Documentation	476
12.166.1.1 HelperMod() [1/2]	476
12.166.1.2 HelperMod() [2/2]	476
12.166.2 Field Documentation	477
12.166.2.1 p	477
12.167 HelperMod< Field, FFLAS::ElementCategories::MachineFloatTag > Struct Template Reference	477
12.167.1 Constructor & Destructor Documentation	477
12.167.1.1 HelperMod() [1/2]	477
12.167.1.2 HelperMod() [2/2]	477
12.167.2 Field Documentation	477
12.167.2.1 p	477
12.167.2.2 invp	477
12.167.2.3 min	477
12.167.2.4 max	477
12.168 Hybrid Struct Reference	478
12.169 Info Struct Reference	478
12.169.1 Constructor & Destructor Documentation	478
12.169.1.1 Info() [1/4]	478
12.169.1.2 Info() [2/4]	478
12.169.1.3 Info() [3/4]	478
12.169.1.4 Info() [4/4]	478
12.169.2 Member Function Documentation	478
12.169.2.1 operator=() [1/2]	478
12.169.2.2 operator=() [2/2]	478
12.169.3 Field Documentation	479
12.169.3.1 size	479
12.169.3.2 perm	479
12.169.3.3 begin	479
12.170 Info Struct Reference	479
12.170.1 Constructor & Destructor Documentation	479
12.170.1.1 Info() [1/4]	479

12.170.1.2 Info() [2/4] . . . . .	479
12.170.1.3 Info() [3/4] . . . . .	479
12.170.1.4 Info() [4/4] . . . . .	479
12.170.2 Member Function Documentation . . . . .	480
12.170.2.1 operator=() [1/2] . . . . .	480
12.170.2.2 operator=() [2/2] . . . . .	480
12.170.3 Field Documentation . . . . .	480
12.170.3.1 size . . . . .	480
12.170.3.2 perm . . . . .	480
12.170.3.3 begin . . . . .	480
12.171 is_all_same< Args > Struct Template Reference . . . . .	480
12.172 is_all_same< T, Args... > Struct Template Reference . . . . .	480
12.172.1 Field Documentation . . . . .	480
12.172.1.1 value . . . . .	480
12.173 is_all_same<> Struct Reference . . . . .	480
12.173.1 Field Documentation . . . . .	481
12.173.1.1 value . . . . .	481
12.174 is_simd< T > Struct Template Reference . . . . .	481
12.174.1 Member Typedef Documentation . . . . .	481
12.174.1.1 type . . . . .	481
12.174.2 Field Documentation . . . . .	481
12.174.2.1 value . . . . .	481
12.175 isSparseMatrix< Field, M > Struct Template Reference . . . . .	481
12.176 isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO > > Struct Template Reference . . . . .	481
12.177 isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO_ZO > > Struct Template Reference . . . . .	482
12.178 isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR > > Struct Template Reference . . . . .	482
12.179 isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_HYB > > Struct Template Reference . . . . .	482
12.180 isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_ZO > > Struct Template Reference . . . . .	483
12.181 isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL > > Struct Template Reference . . . . .	483
12.182 isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd > > Struct Template Reference . . . . .	483
12.183 isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > > Struct Template Reference . . . . .	484
12.184 isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_ZO > > Struct Template Reference . . . . .	484
12.185 isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::HYB_ZO > > Struct Template Reference . . . . .	484
12.186 isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL > > Struct Template Reference . . . . .	484
12.187 isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL_ZO > > Struct Template Reference . . . . .	485
12.188 isSparseMatrixMKLFormat< F, M > Struct Template Reference . . . . .	485
12.189 isSparseMatrixSimdFormat< F, M > Struct Template Reference . . . . .	485
12.190 isZOSparseMatrix< F, M > Struct Template Reference . . . . .	486
12.191 isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO_ZO > > Struct Template Reference . . . . .	486
12.192 isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_ZO > > Struct Template Reference . . . . .	486

12.193 isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > > Struct Template Reference . . . . .	486
12.194 isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_ZO > > Struct Template Reference . . . . .	487
12.195 isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL_ZO > > Struct Template Reference . . . . .	487
12.196 Iterative Struct Reference . . . . .	487
12.197 LazyTag Struct Reference . . . . .	487
12.197.1 Detailed Description . . . . .	488
12.198 limits< T > Struct Template Reference . . . . .	488
12.199 limits< char > Struct Reference . . . . .	488
12.199.1 Member Typedef Documentation . . . . .	488
12.199.1.1 T . . . . .	488
12.199.2 Member Function Documentation . . . . .	488
12.199.2.1 max() . . . . .	488
12.199.2.2 min() . . . . .	488
12.199.2.3 digits() . . . . .	488
12.200 limits< double > Struct Reference . . . . .	488
12.200.1 Member Typedef Documentation . . . . .	489
12.200.1.1 T . . . . .	489
12.200.2 Member Function Documentation . . . . .	489
12.200.2.1 max() . . . . .	489
12.200.2.2 min() . . . . .	489
12.200.2.3 digits() . . . . .	489
12.201 limits< float > Struct Reference . . . . .	489
12.201.1 Member Typedef Documentation . . . . .	489
12.201.1.1 T . . . . .	489
12.201.2 Member Function Documentation . . . . .	489
12.201.2.1 max() . . . . .	489
12.201.2.2 min() . . . . .	489
12.201.2.3 digits() . . . . .	490
12.202 limits< Givaro::Integer > Struct Reference . . . . .	490
12.202.1 Member Typedef Documentation . . . . .	490
12.202.1.1 T . . . . .	490
12.202.2 Member Function Documentation . . . . .	490
12.202.2.1 max() . . . . .	490
12.202.2.2 min() . . . . .	490
12.203 limits< int > Struct Reference . . . . .	490
12.203.1 Member Typedef Documentation . . . . .	490
12.203.1.1 T . . . . .	490
12.203.2 Member Function Documentation . . . . .	491
12.203.2.1 max() . . . . .	491
12.203.2.2 min() . . . . .	491
12.203.2.3 digits() . . . . .	491

12.204 limits< long > Struct Reference . . . . .	491
12.204.1 Member Typedef Documentation . . . . .	491
12.204.1.1 T . . . . .	491
12.204.2 Member Function Documentation . . . . .	491
12.204.2.1 max() . . . . .	491
12.204.2.2 min() . . . . .	491
12.204.2.3 digits() . . . . .	491
12.205 limits< long long > Struct Reference . . . . .	491
12.205.1 Member Typedef Documentation . . . . .	492
12.205.1.1 T . . . . .	492
12.205.2 Member Function Documentation . . . . .	492
12.205.2.1 max() . . . . .	492
12.205.2.2 min() . . . . .	492
12.205.2.3 digits() . . . . .	492
12.206 limits< Reclnt::rint< K > > Struct Template Reference . . . . .	492
12.206.1 Member Typedef Documentation . . . . .	492
12.206.1.1 T . . . . .	492
12.206.2 Member Function Documentation . . . . .	492
12.206.2.1 max() . . . . .	492
12.206.2.2 min() . . . . .	493
12.207 limits< Reclnt::ruint< K > > Struct Template Reference . . . . .	493
12.207.1 Member Typedef Documentation . . . . .	493
12.207.1.1 T . . . . .	493
12.207.2 Member Function Documentation . . . . .	493
12.207.2.1 max() . . . . .	493
12.207.2.2 min() . . . . .	493
12.208 limits< short int > Struct Reference . . . . .	493
12.208.1 Member Typedef Documentation . . . . .	494
12.208.1.1 T . . . . .	494
12.208.2 Member Function Documentation . . . . .	494
12.208.2.1 max() . . . . .	494
12.208.2.2 min() . . . . .	494
12.208.2.3 digits() . . . . .	494
12.209 limits< signed char > Struct Reference . . . . .	494
12.209.1 Member Typedef Documentation . . . . .	494
12.209.1.1 T . . . . .	494
12.209.2 Member Function Documentation . . . . .	494
12.209.2.1 max() . . . . .	494
12.209.2.2 min() . . . . .	494
12.209.2.3 digits() . . . . .	494
12.210 limits< unsigned char > Struct Reference . . . . .	495
12.210.1 Member Typedef Documentation . . . . .	495

12.210.1.1 T	495
12.210.2 Member Function Documentation	495
12.210.2.1 max()	495
12.210.2.2 min()	495
12.210.2.3 digits()	495
12.211 limits< unsigned int > Struct Reference	495
12.211.1 Member Typedef Documentation	495
12.211.1.1 T	495
12.211.2 Member Function Documentation	496
12.211.2.1 max()	496
12.211.2.2 min()	496
12.211.2.3 digits()	496
12.212 limits< unsigned long > Struct Reference	496
12.212.1 Member Typedef Documentation	496
12.212.1.1 T	496
12.212.2 Member Function Documentation	496
12.212.2.1 max()	496
12.212.2.2 min()	496
12.212.2.3 digits()	496
12.213 limits< unsigned long long > Struct Reference	496
12.213.1 Member Typedef Documentation	497
12.213.1.1 T	497
12.213.2 Member Function Documentation	497
12.213.2.1 max()	497
12.213.2.2 min()	497
12.213.2.3 digits()	497
12.214 limits< unsigned short int > Struct Reference	497
12.214.1 Member Typedef Documentation	497
12.214.1.1 T	497
12.214.2 Member Function Documentation	497
12.214.2.1 max()	497
12.214.2.2 min()	497
12.214.2.3 digits()	498
12.215 MachineFloatTag Struct Reference	498
12.215.1 Detailed Description	498
12.216 MachineIntTag Struct Reference	498
12.216.1 Detailed Description	498
12.217 MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait > Struct Template Reference	498
12.217.1 Member Typedef Documentation	499
12.217.1.1 Self_t	499
12.217.1.2 DelayedField_t	499
12.217.1.3 DelayedField	499



12.217.1.4 DFElt	499
12.217.2 Constructor & Destructor Documentation	499
12.217.2.1 MMHelper() [1/5]	499
12.217.2.2 MMHelper() [2/5]	500
12.217.2.3 MMHelper() [3/5]	500
12.217.2.4 MMHelper() [4/5]	500
12.217.2.5 MMHelper() [5/5]	500
12.217.3 Member Function Documentation	500
12.217.3.1 initC()	500
12.217.3.2 initA()	500
12.217.3.3 initB()	500
12.217.3.4 initOut()	500
12.217.3.5 MaxDelayedDim()	501
12.217.3.6 Aunfit()	501
12.217.3.7 Bunfit()	501
12.217.3.8 setOutBounds()	501
12.217.3.9 checkA()	501
12.217.3.10 checkB()	501
12.217.3.11 checkOut() [1/2]	501
12.217.3.12 checkOut() [2/2]	501
12.217.4 Friends And Related Symbol Documentation	502
12.217.4.1 operator<<	502
12.217.5 Field Documentation	502
12.217.5.1 recLevel	502
12.217.5.2 FieldMin	502
12.217.5.3 FieldMax	502
12.217.5.4 Amin	502
12.217.5.5 Amax	502
12.217.5.6 Bmin	502
12.217.5.7 Bmax	502
12.217.5.8 Cmin	502
12.217.5.9 Cmax	503
12.217.5.10 Outmin	503
12.217.5.11 Outmax	503
12.217.5.12 MaxStorableValue	503
12.217.5.13 delayedField	503
12.217.5.14 parseq	503
12.218 MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait > Struct Template Reference	503
12.218.1 Member Typedef Documentation	504
12.218.1.1 Self_t	504
12.218.2 Constructor & Destructor Documentation	504

12.218.2.1 MMHelper() [1/5]	504
12.218.2.2 MMHelper() [2/5]	504
12.218.2.3 MMHelper() [3/5]	504
12.218.2.4 MMHelper() [4/5]	504
12.218.2.5 MMHelper() [5/5]	504
12.218.3 Member Function Documentation	504
12.218.3.1 setNorm()	504
12.218.4 Friends And Related Symbol Documentation	505
12.218.4.1 operator<<	505
12.218.5 Field Documentation	505
12.218.5.1 normA	505
12.218.5.2 normB	505
12.218.5.3 recLevel	505
12.218.5.4 parseq	505
12.219 MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeq↵ Trait > Struct Template Reference	505
12.219.1 Member Typedef Documentation	506
12.219.1.1 Self_t	506
12.219.2 Constructor & Destructor Documentation	506
12.219.2.1 MMHelper() [1/5]	506
12.219.2.2 MMHelper() [2/5]	506
12.219.2.3 MMHelper() [3/5]	506
12.219.2.4 MMHelper() [4/5]	506
12.219.2.5 MMHelper() [5/5]	506
12.219.3 Member Function Documentation	506
12.219.3.1 setNorm()	506
12.219.4 Friends And Related Symbol Documentation	507
12.219.4.1 operator<<	507
12.219.5 Field Documentation	507
12.219.5.1 normA	507
12.219.5.2 normB	507
12.219.5.3 recLevel	507
12.219.5.4 parseq	507
12.220 MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< Dest >, ParSeqTrait > Struct Tem- plate Reference	507
12.220.1 Member Typedef Documentation	508
12.220.1.1 Self_t	508
12.220.2 Constructor & Destructor Documentation	508
12.220.2.1 MMHelper() [1/4]	508
12.220.2.2 MMHelper() [2/4]	508
12.220.2.3 MMHelper() [3/4]	508
12.220.2.4 MMHelper() [4/4]	508
12.220.3 Friends And Related Symbol Documentation	508

12.220.3.1 operator<<	508
12.220.4 Field Documentation	508
12.220.4.1 recLevel	508
12.220.4.2 parseq	508
12.221 MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeqTrait > Struct Template Reference	509
12.221.1 Member Typedef Documentation	509
12.221.1.1 Self_t	509
12.221.2 Constructor & Destructor Documentation	509
12.221.2.1 MMHelper() [1/5]	509
12.221.2.2 MMHelper() [2/5]	509
12.221.2.3 MMHelper() [3/5]	509
12.221.2.4 MMHelper() [4/5]	510
12.221.2.5 MMHelper() [5/5]	510
12.221.3 Member Function Documentation	510
12.221.3.1 setNorm()	510
12.221.4 Friends And Related Symbol Documentation	510
12.221.4.1 operator<<	510
12.221.5 Field Documentation	510
12.221.5.1 normA	510
12.221.5.2 normB	510
12.221.5.3 recLevel	510
12.221.5.4 parseq	510
12.222 MMHelper< Field, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait > Struct Template Reference	511
12.222.1 Detailed Description	511
12.222.2 Member Typedef Documentation	511
12.222.2.1 Self_t	511
12.222.3 Constructor & Destructor Documentation	511
12.222.3.1 MMHelper() [1/4]	511
12.222.3.2 MMHelper() [2/4]	511
12.222.3.3 MMHelper() [3/4]	512
12.222.3.4 MMHelper() [4/4]	512
12.222.4 Friends And Related Symbol Documentation	512
12.222.4.1 operator<<	512
12.222.5 Field Documentation	512
12.222.5.1 recLevel	512
12.222.5.2 parseq	512
12.223 ModeTraits< Field > Struct Template Reference	512
12.223.1 Detailed Description	512
12.223.2 Member Typedef Documentation	512
12.223.2.1 value	512
12.224 ModeTraits< Givaro::Modular< Element, Compute > > Struct Template Reference	513

12.224.1 Member Typedef Documentation	513
12.224.1.1 value	513
12.225 ModeTraits< Givaro::Modular< Givaro::Integer, Compute > > Struct Template Reference	513
12.225.1 Member Typedef Documentation	513
12.225.1.1 value	513
12.226 ModeTraits< Givaro::Modular< int16_t, Compute > > Struct Template Reference	513
12.226.1 Member Typedef Documentation	513
12.226.1.1 value	513
12.227 ModeTraits< Givaro::Modular< int32_t, Compute > > Struct Template Reference	514
12.227.1 Member Typedef Documentation	514
12.227.1.1 value	514
12.228 ModeTraits< Givaro::Modular< int64_t, uint64_t > > Struct Reference	514
12.228.1 Member Typedef Documentation	514
12.228.1.1 value	514
12.229 ModeTraits< Givaro::Modular< int8_t, Compute > > Struct Template Reference	514
12.229.1 Member Typedef Documentation	514
12.229.1.1 value	514
12.230 ModeTraits< Givaro::Modular< RecInt::ruint< K >, Compute > > Struct Template Reference	515
12.230.1 Member Typedef Documentation	515
12.230.1.1 value	515
12.231 ModeTraits< Givaro::Modular< uint16_t, Compute > > Struct Template Reference	515
12.231.1 Member Typedef Documentation	515
12.231.1.1 value	515
12.232 ModeTraits< Givaro::Modular< uint32_t, Compute > > Struct Template Reference	515
12.232.1 Member Typedef Documentation	515
12.232.1.1 value	515
12.233 ModeTraits< Givaro::Modular< uint8_t, Compute > > Struct Template Reference	516
12.233.1 Member Typedef Documentation	516
12.233.1.1 value	516
12.234 ModeTraits< Givaro::ModularBalanced< Element > > Struct Template Reference	516
12.234.1 Member Typedef Documentation	516
12.234.1.1 value	516
12.235 ModeTraits< Givaro::ModularBalanced< Givaro::Integer > > Struct Reference	516
12.235.1 Member Typedef Documentation	516
12.235.1.1 value	516
12.236 ModeTraits< Givaro::ModularBalanced< int16_t > > Struct Reference	517
12.236.1 Member Typedef Documentation	517
12.236.1.1 value	517
12.237 ModeTraits< Givaro::ModularBalanced< int32_t > > Struct Reference	517
12.237.1 Member Typedef Documentation	517
12.237.1.1 value	517
12.238 ModeTraits< Givaro::ModularBalanced< int8_t > > Struct Reference	517

12.238.1 Member Typedef Documentation	517
12.238.1.1 value	517
12.239 ModeTraits< Givaro::Montgomery< T > > Struct Template Reference	517
12.239.1 Member Typedef Documentation	518
12.239.1.1 value	518
12.240 ModeTraits< Givaro::ZRing< double > > Struct Reference	518
12.240.1 Member Typedef Documentation	518
12.240.1.1 value	518
12.241 ModeTraits< Givaro::ZRing< float > > Struct Reference	518
12.241.1 Member Typedef Documentation	518
12.241.1.1 value	518
12.242 ModeTraits< Givaro::ZRing< Givaro::Integer > > Struct Reference	518
12.242.1 Member Typedef Documentation	519
12.242.1.1 value	519
12.243 ModularBalanced< T > Class Template Reference	519
12.244 ModularTag Struct Reference	519
12.244.1 Detailed Description	519
12.245 Montgomery< T > Class Template Reference	519
12.246 need_field_characteristic< Field > Struct Template Reference	519
12.246.1 Field Documentation	519
12.246.1.1 value	519
12.247 need_field_characteristic< Givaro::Modular< Field > > Struct Template Reference	519
12.247.1 Field Documentation	520
12.247.1.1 value	520
12.248 need_field_characteristic< Givaro::ModularBalanced< Field > > Struct Template Reference	520
12.248.1 Field Documentation	520
12.248.1.1 value	520
12.249 NoSimd< T > Struct Template Reference	520
12.249.1 Member Typedef Documentation	520
12.249.1.1 vect_t	520
12.249.1.2 scalar_t	521
12.249.1.3 aligned_allocator	521
12.249.1.4 aligned_vector	521
12.249.1.5 is_same_element	521
12.249.2 Member Function Documentation	521
12.249.2.1 type_string()	521
12.249.2.2 valid()	521
12.249.2.3 compliant()	521
12.249.3 Field Documentation	521
12.249.3.1 vect_size	521
12.249.3.2 alignment	521
12.250 Parallel< C, P > Struct Template Reference	522

12.250.1 Member Typedef Documentation	522
12.250.1.1 Cut	522
12.250.1.2 Param	522
12.250.2 Constructor & Destructor Documentation	522
12.250.2.1 Parallel()	522
12.250.3 Member Function Documentation	522
12.250.3.1 numthreads()	522
12.250.3.2 set_numthreads()	522
12.250.4 Friends And Related Symbol Documentation	522
12.250.4.1 operator<<	522
12.251 RNSInteger< RNS >::RandIter Class Reference	523
12.251.1 Constructor & Destructor Documentation	523
12.251.1.1 RandIter()	523
12.251.2 Member Function Documentation	523
12.251.2.1 random() [1/2]	523
12.251.2.2 random() [2/2]	523
12.251.2.3 operator>() [1/2]	523
12.251.2.4 operator>() [2/2]	523
12.251.2.5 ring()	524
12.252 RNSIntegerMod< RNS >::RandIter Class Reference	524
12.252.1 Constructor & Destructor Documentation	524
12.252.1.1 RandIter()	524
12.252.2 Member Function Documentation	524
12.252.2.1 random() [1/2]	524
12.252.2.2 random() [2/2]	524
12.252.2.3 operator>() [1/2]	524
12.252.2.4 operator>() [2/2]	524
12.252.2.5 ring()	525
12.253 readMyMachineType< Field, T > Struct Template Reference	525
12.253.1 Member Typedef Documentation	525
12.253.1.1 Element	525
12.253.1.2 Element_ptr	525
12.253.2 Member Function Documentation	525
12.253.2.1 operator>()	525
12.254 readMyMachineType< Field, mpz_t > Struct Template Reference	525
12.254.1 Member Typedef Documentation	526
12.254.1.1 Element	526
12.254.1.2 Element_ptr	526
12.254.2 Member Function Documentation	526
12.254.2.1 operator>()	526
12.255 Recursive Struct Reference	526
12.256 Recursive Struct Reference	526

12.257 rint< K > Class Template Reference . . . . .	526
12.258 rns_double Struct Reference . . . . .	526
12.258.1 Member Typedef Documentation . . . . .	527
12.258.1.1 integer . . . . .	527
12.258.1.2 ModField . . . . .	528
12.258.1.3 BasisElement . . . . .	528
12.258.1.4 Element . . . . .	528
12.258.1.5 Element_ptr . . . . .	528
12.258.1.6 ConstElement_ptr . . . . .	528
12.258.2 Constructor & Destructor Documentation . . . . .	528
12.258.2.1 rns_double() [1/4] . . . . .	528
12.258.2.2 rns_double() [2/4] . . . . .	528
12.258.2.3 rns_double() [3/4] . . . . .	528
12.258.2.4 rns_double() [4/4] . . . . .	528
12.258.3 Member Function Documentation . . . . .	528
12.258.3.1 precompute_cst() . . . . .	528
12.258.3.2 init() [1/3] . . . . .	529
12.258.3.3 init() [2/3] . . . . .	529
12.258.3.4 init_transpose() . . . . .	529
12.258.3.5 convert() [1/2] . . . . .	529
12.258.3.6 convert_transpose() . . . . .	529
12.258.3.7 reduce() . . . . .	530
12.258.3.8 init() [3/3] . . . . .	530
12.258.3.9 convert() [2/2] . . . . .	530
12.258.4 Field Documentation . . . . .	530
12.258.4.1 _basis . . . . .	530
12.258.4.2 _basisMax . . . . .	530
12.258.4.3 _negbasis . . . . .	530
12.258.4.4 _invbasis . . . . .	530
12.258.4.5 _field_rns . . . . .	530
12.258.4.6 _M . . . . .	530
12.258.4.7 _Mi . . . . .	531
12.258.4.8 _MMi . . . . .	531
12.258.4.9 _crt_in . . . . .	531
12.258.4.10 _crt_out . . . . .	531
12.258.4.11 _size . . . . .	531
12.258.4.12 _pbits . . . . .	531
12.258.4.13 _ldm . . . . .	531
12.258.4.14 _mi_sum . . . . .	531
12.259 rns_double_elt Struct Reference . . . . .	531
12.259.1 Constructor & Destructor Documentation . . . . .	532
12.259.1.1 rns_double_elt() [1/3] . . . . .	532

12.259.1.2 <code>~rns_double_elt()</code>	532
12.259.1.3 <code>rns_double_elt()</code> [2/3]	532
12.259.1.4 <code>rns_double_elt()</code> [3/3]	532
12.259.2 Member Function Documentation	532
12.259.2.1 <code>operator&amp;()</code> [1/2]	532
12.259.2.2 <code>operator&amp;()</code> [2/2]	532
12.259.3 Field Documentation	532
12.259.3.1 <code>_ptr</code>	532
12.259.3.2 <code>_stride</code>	532
12.259.3.3 <code>_alloc</code>	532
12.260 <code>rns_double_elt_cstptr</code> Struct Reference	533
12.260.1 Constructor & Destructor Documentation	533
12.260.1.1 <code>rns_double_elt_cstptr()</code> [1/5]	533
12.260.1.2 <code>rns_double_elt_cstptr()</code> [2/5]	533
12.260.1.3 <code>rns_double_elt_cstptr()</code> [3/5]	533
12.260.1.4 <code>rns_double_elt_cstptr()</code> [4/5]	534
12.260.1.5 <code>rns_double_elt_cstptr()</code> [5/5]	534
12.260.2 Member Function Documentation	534
12.260.2.1 <code>operator&amp;()</code> [1/2]	534
12.260.2.2 <code>operator*()</code>	534
12.260.2.3 <code>operator[]()</code> [1/2]	534
12.260.2.4 <code>operator[]()</code> [2/2]	534
12.260.2.5 <code>operator++()</code>	534
12.260.2.6 <code>operator--()</code>	534
12.260.2.7 <code>operator+()</code>	534
12.260.2.8 <code>operator-()</code>	534
12.260.2.9 <code>operator+=()</code>	534
12.260.2.10 <code>operator-=()</code>	534
12.260.2.11 <code>operator=()</code>	535
12.260.2.12 <code>operator&lt;()</code>	535
12.260.2.13 <code>operator"!=()</code>	535
12.260.2.14 <code>operator&amp;()</code> [2/2]	535
12.260.3 Field Documentation	535
12.260.3.1 <code>other</code>	535
12.260.3.2 <code>_ptr</code>	535
12.260.3.3 <code>_stride</code>	535
12.260.3.4 <code>_alloc</code>	535
12.261 <code>rns_double_elt_ptr</code> Struct Reference	535
12.261.1 Constructor & Destructor Documentation	536
12.261.1.1 <code>rns_double_elt_ptr()</code> [1/5]	536
12.261.1.2 <code>rns_double_elt_ptr()</code> [2/5]	536
12.261.1.3 <code>rns_double_elt_ptr()</code> [3/5]	536



12.261.1.4 rns_double_elt_ptr() [4/5]	536
12.261.1.5 rns_double_elt_ptr() [5/5]	536
12.261.2 Member Function Documentation	536
12.261.2.1 operator&() [1/2]	536
12.261.2.2 operator*()	536
12.261.2.3 operator[]() [1/2]	537
12.261.2.4 operator[]() [2/2]	537
12.261.2.5 operator++()	537
12.261.2.6 operator--()	537
12.261.2.7 operator+()	537
12.261.2.8 operator-()	537
12.261.2.9 operator+=()	537
12.261.2.10 operator-=()	537
12.261.2.11 operator=()	537
12.261.2.12 operator<()	537
12.261.2.13 operator"!="()	537
12.261.2.14 operator&() [2/2]	537
12.261.3 Field Documentation	538
12.261.3.1 other	538
12.261.3.2 _ptr	538
12.261.3.3 _stride	538
12.261.3.4 _alloc	538
12.262 rns_double_extended Struct Reference	538
12.262.1 Member Typedef Documentation	539
12.262.1.1 integer	539
12.262.1.2 ModField	539
12.262.1.3 BasisElement	539
12.262.1.4 Element	539
12.262.1.5 Element_ptr	539
12.262.1.6 ConstElement_ptr	539
12.262.2 Constructor & Destructor Documentation	539
12.262.2.1 rns_double_extended() [1/3]	539
12.262.2.2 rns_double_extended() [2/3]	539
12.262.2.3 rns_double_extended() [3/3]	539
12.262.3 Member Function Documentation	540
12.262.3.1 precompute_cst()	540
12.262.3.2 init() [1/3]	540
12.262.3.3 init() [2/3]	540
12.262.3.4 convert() [1/2]	540
12.262.3.5 init() [3/3]	540
12.262.3.6 convert() [2/2]	540
12.262.3.7 reduce()	540

12.262.4 Field Documentation	541
12.262.4.1 _basis	541
12.262.4.2 _basisMax	541
12.262.4.3 _negbasis	541
12.262.4.4 _invbasis	541
12.262.4.5 _field_rns	541
12.262.4.6 _M	541
12.262.4.7 _Mi	541
12.262.4.8 _MMi	541
12.262.4.9 _crt_in	541
12.262.4.10 _crt_out	541
12.262.4.11 _size	541
12.262.4.12 _pbits	541
12.262.4.13 _ldm	541
12.263 RNSElementTag Struct Reference	542
12.263.1 Detailed Description	542
12.264 RNSInteger< RNS > Class Template Reference	542
12.264.1 Member Typedef Documentation	543
12.264.1.1 BasisElement	543
12.264.1.2 integer	543
12.264.1.3 Element	543
12.264.1.4 Element_ptr	543
12.264.1.5 ConstElement_ptr	543
12.264.2 Constructor & Destructor Documentation	543
12.264.2.1 RNSInteger() [1/2]	543
12.264.2.2 RNSInteger() [2/2]	543
12.264.3 Member Function Documentation	543
12.264.3.1 rns()	543
12.264.3.2 size()	543
12.264.3.3 isOne()	544
12.264.3.4 isMOne()	544
12.264.3.5 isZero()	544
12.264.3.6 characteristic()	544
12.264.3.7 cardinality()	544
12.264.3.8 init() [1/2]	544
12.264.3.9 init() [2/2]	544
12.264.3.10 reduce() [1/2]	544
12.264.3.11 reduce() [2/2]	544
12.264.3.12 convert()	545
12.264.3.13 assign()	545
12.264.3.14 write() [1/2]	545
12.264.3.15 write() [2/2]	545

12.264.4 Field Documentation	545
12.264.4.1 _rns	545
12.264.4.2 one	545
12.264.4.3 mOne	545
12.264.4.4 zero	545
12.265 RNSIntegerMod< RNS > Class Template Reference	545
12.265.1 Member Typedef Documentation	547
12.265.1.1 Element	547
12.265.1.2 Element_ptr	547
12.265.1.3 ConstElement_ptr	547
12.265.1.4 BasisElement	547
12.265.1.5 ModField	547
12.265.1.6 integer	547
12.265.2 Constructor & Destructor Documentation	547
12.265.2.1 RNSIntegerMod()	547
12.265.3 Member Function Documentation	547
12.265.3.1 rns()	547
12.265.3.2 delayed()	547
12.265.3.3 size()	548
12.265.3.4 isOne()	548
12.265.3.5 isMOne()	548
12.265.3.6 isZero()	548
12.265.3.7 characteristic() [1/2]	548
12.265.3.8 characteristic() [2/2]	548
12.265.3.9 cardinality() [1/2]	548
12.265.3.10 cardinality() [2/2]	548
12.265.3.11 minElement()	548
12.265.3.12 maxElement()	548
12.265.3.13 init() [1/3]	548
12.265.3.14 init() [2/3]	549
12.265.3.15 reduce() [1/2]	549
12.265.3.16 reduce() [2/2]	549
12.265.3.17 init() [3/3]	549
12.265.3.18 convert()	549
12.265.3.19 assign()	549
12.265.3.20 add()	549
12.265.3.21 sub()	549
12.265.3.22 neg()	550
12.265.3.23 mul()	550
12.265.3.24 axpyin()	550
12.265.3.25 inv()	550
12.265.3.26 areEqual()	550

12.265.3.27 write() [1/2]	550
12.265.3.28 write() [2/2]	550
12.265.3.29 reduce_modp() [1/2]	550
12.265.3.30 write_matrix()	551
12.265.3.31 write_matrix_long()	551
12.265.3.32 reduce_modp() [2/2]	551
12.265.3.33 reduce_modp_rnsmajor()	551
12.265.4 Field Documentation	551
12.265.4.1 _p	551
12.265.4.2 _Mi_modp_rns	551
12.265.4.3 _iM_modp_rns	551
12.265.4.4 _rns	551
12.265.4.5 _F	552
12.265.4.6 _RNSdelayed	552
12.265.4.7 one	552
12.265.4.8 mOne	552
12.265.4.9 zero	552
12.266 rnsRandIter< RNS > Class Template Reference	552
12.266.1 Constructor & Destructor Documentation	552
12.266.1.1 rnsRandIter()	552
12.266.2 Member Function Documentation	553
12.266.2.1 random() [1/2]	553
12.266.2.2 operator>() [1/2]	553
12.266.2.3 operator>() [2/2]	553
12.266.2.4 random() [2/2]	553
12.266.2.5 ring()	553
12.267 Row Struct Reference	553
12.268 ruint< K > Class Template Reference	553
12.269 ScalFunctions< Element > Struct Template Reference	553
12.269.1 Member Typedef Documentation	554
12.269.1.1 vectElt	554
12.269.2 Member Function Documentation	554
12.269.2.1 genInputs()	554
12.269.2.2 genInputsWithZero()	555
12.269.2.3 zero()	555
12.269.2.4 vand()	555
12.269.2.5 vor()	555
12.269.2.6 vxor()	555
12.269.2.7 vandnot()	555
12.269.2.8 add()	555
12.269.2.9 addin()	555
12.269.2.10 sub()	555

12.269.2.11	subin()	556
12.269.2.12	mul()	556
12.269.2.13	mulin()	556
12.269.2.14	div()	556
12.269.2.15	fmadd()	556
12.269.2.16	fmaddin()	556
12.269.2.17	fmsub()	556
12.269.2.18	fmsubin()	556
12.269.2.19	fnmadd()	557
12.269.2.20	fnmaddin()	557
12.269.2.21	lesser()	557
12.269.2.22	lesser_eq()	557
12.269.2.23	greater()	557
12.269.2.24	greater_eq()	557
12.269.2.25	eq()	557
12.269.2.26	unpacklo()	557
12.269.2.27	unpackhi()	558
12.269.2.28	unpacklohi()	558
12.269.2.29	pack_even()	558
12.269.2.30	pack_odd()	558
12.269.2.31	pack()	558
12.269.2.32	blend()	558
12.270	ScalFunctionsBase< Element, Enable > Struct Template Reference	558
12.271	ScalFunctionsBase< Element, typename enable_if< is_floating_point< Element >::value >::type > Struct Template Reference	559
12.271.1	Member Function Documentation	559
12.271.1.1	get_default_random_generator()	559
12.271.1.2	ceil()	559
12.271.1.3	floor()	559
12.271.1.4	round()	559
12.271.1.5	blendv()	560
12.271.1.6	fma()	560
12.271.2	Field Documentation	560
12.271.2.1	_zero	560
12.271.2.2	cmp_true	560
12.271.2.3	cmp_false	560
12.272	ScalFunctionsBase< Element, typename enable_if< is_integral< Element >::value >::type > Struct Template Reference	560
12.272.1	Member Function Documentation	561
12.272.1.1	get_default_random_generator()	561
12.272.1.2	round()	561
12.272.1.3	fma()	561
12.272.1.4	mullo()	561

12.272.1.5 mulhi()	561
12.272.1.6 mulx()	561
12.272.1.7 fmaddx()	561
12.272.1.8 fmaddxin()	562
12.272.1.9 fmsubx()	562
12.272.1.10 fmsubxin()	562
12.272.1.11 fnmaddx()	562
12.272.1.12 fnmaddxin()	562
12.272.1.13 sra()	562
12.272.1.14 srl()	562
12.272.1.15 sll()	562
12.272.2 Field Documentation	563
12.272.2.1 _zero	563
12.272.2.2 cmp_true	563
12.272.2.3 cmp_false	563
12.273 Sequential Struct Reference	563
12.273.1 Constructor & Destructor Documentation	563
12.273.1.1 Sequential() [1/3]	563
12.273.1.2 Sequential() [2/3]	563
12.273.1.3 Sequential() [3/3]	563
12.273.2 Member Function Documentation	563
12.273.2.1 numthreads()	563
12.273.3 Friends And Related Symbol Documentation	564
12.273.3.1 operator<<	564
12.274 Simd128_impl< ArithType, Int, Signed, Size > Struct Template Reference	564
12.275 Simd128_impl< true, false, true, 4 > Struct Reference	564
12.276 Simd128_impl< true, false, true, 8 > Struct Reference	564
12.277 Simd128_impl< true, true, false, 2 > Struct Reference	564
12.277.1 Member Typedef Documentation	566
12.277.1.1 scalar_t	566
12.277.1.2 aligned_allocator	566
12.277.1.3 aligned_vector	566
12.277.1.4 is_same_element	566
12.277.1.5 vect_t	566
12.277.2 Member Function Documentation	567
12.277.2.1 type_string()	567
12.277.2.2 set1() [1/2]	567
12.277.2.3 set() [1/2]	567
12.277.2.4 gather() [1/2]	567
12.277.2.5 load() [1/2]	567
12.277.2.6 loadu() [1/2]	567
12.277.2.7 store() [1/2]	567

12.277.2.8 storeu() [1/2]	567
12.277.2.9 stream() [1/2]	567
12.277.2.10 sra()	568
12.277.2.11 greater()	568
12.277.2.12 lesser()	568
12.277.2.13 greater_eq()	568
12.277.2.14 lesser_eq()	568
12.277.2.15 mulhi()	568
12.277.2.16 mulx()	568
12.277.2.17 fmaddx()	568
12.277.2.18 fmaddxin()	568
12.277.2.19 fnmaddx()	569
12.277.2.20 fnmaddxin()	569
12.277.2.21 fmsubx()	569
12.277.2.22 fmsubxin()	569
12.277.2.23 hadd_to_scal()	569
12.277.2.24 valid()	569
12.277.2.25 compliant()	569
12.277.2.26 set1() [2/2]	569
12.277.2.27 set() [2/2]	569
12.277.2.28 gather() [2/2]	570
12.277.2.29 load() [2/2]	570
12.277.2.30 loadu() [2/2]	570
12.277.2.31 store() [2/2]	570
12.277.2.32 storeu() [2/2]	570
12.277.2.33 stream() [2/2]	570
12.277.2.34 sll()	570
12.277.2.35 srl()	570
12.277.2.36 shuffle()	570
12.277.2.37 unpacklo_intrinsic()	571
12.277.2.38 unpackhi_intrinsic()	571
12.277.2.39 unpacklo()	571
12.277.2.40 unpackhi()	571
12.277.2.41 unpacklohi()	571
12.277.2.42 pack_even()	571
12.277.2.43 pack_odd()	571
12.277.2.44 pack()	571
12.277.2.45 transpose()	571
12.277.2.46 blend()	572
12.277.2.47 add()	572
12.277.2.48 addin()	572
12.277.2.49 sub()	572

12.277.2.50 subin()	572
12.277.2.51 mullo()	572
12.277.2.52 mul()	572
12.277.2.53 fmadd()	572
12.277.2.54 fmaddin()	572
12.277.2.55 fnmadd()	573
12.277.2.56 fnmaddin()	573
12.277.2.57 fmsub()	573
12.277.2.58 fmsubin()	573
12.277.2.59 eq()	573
12.277.2.60 round()	573
12.277.2.61 mod()	573
12.277.2.62 zero()	573
12.277.2.63 sll128()	573
12.277.2.64 srl128()	574
12.277.2.65 vand()	574
12.277.2.66 vor()	574
12.277.2.67 vxor()	574
12.277.2.68 vandnot()	574
12.277.3 Field Documentation	574
12.277.3.1 vect_size	574
12.277.3.2 alignment	574
12.278 Simd128_impl< true, true, false, 4 > Struct Reference	574
12.278.1 Member Typedef Documentation	576
12.278.1.1 scalar_t	576
12.278.1.2 aligned_allocator	576
12.278.1.3 aligned_vector	576
12.278.1.4 is_same_element	577
12.278.1.5 vect_t	577
12.278.2 Member Function Documentation	577
12.278.2.1 type_string()	577
12.278.2.2 set1() [1/2]	577
12.278.2.3 set() [1/2]	577
12.278.2.4 gather() [1/2]	577
12.278.2.5 load() [1/2]	577
12.278.2.6 loadu() [1/2]	577
12.278.2.7 store() [1/2]	577
12.278.2.8 storeu() [1/2]	577
12.278.2.9 stream() [1/2]	578
12.278.2.10 sra()	578
12.278.2.11 greater()	578
12.278.2.12 lesser()	578



12.278.2.13 greater_eq()	578
12.278.2.14 lesser_eq()	578
12.278.2.15 mulhi()	578
12.278.2.16 mulx()	578
12.278.2.17 fmaddx()	578
12.278.2.18 fmaddxin()	578
12.278.2.19 fnmaddx()	579
12.278.2.20 fnmaddxin()	579
12.278.2.21 fmsubx()	579
12.278.2.22 fmsubxin()	579
12.278.2.23 hadd_to_scal()	579
12.278.2.24 valid()	579
12.278.2.25 compliant()	579
12.278.2.26 set1() [2/2]	579
12.278.2.27 set() [2/2]	579
12.278.2.28 gather() [2/2]	580
12.278.2.29 load() [2/2]	580
12.278.2.30 loadu() [2/2]	580
12.278.2.31 store() [2/2]	580
12.278.2.32 storeu() [2/2]	580
12.278.2.33 stream() [2/2]	580
12.278.2.34 sll()	580
12.278.2.35 srl()	580
12.278.2.36 shuffle()	580
12.278.2.37 unpacklo_intrinsic()	580
12.278.2.38 unpackhi_intrinsic()	581
12.278.2.39 unpacklo()	581
12.278.2.40 unpackhi()	581
12.278.2.41 unpacklohi()	581
12.278.2.42 pack_even()	581
12.278.2.43 pack_odd()	581
12.278.2.44 pack()	581
12.278.2.45 transpose()	581
12.278.2.46 blend()	581
12.278.2.47 add()	582
12.278.2.48 addin()	582
12.278.2.49 sub()	582
12.278.2.50 subin()	582
12.278.2.51 mullo()	582
12.278.2.52 mul()	582
12.278.2.53 fmadd()	582
12.278.2.54 fmaddin()	582

12.278.2.55 fnmadd()	582
12.278.2.56 fnmaddin()	583
12.278.2.57 fmsub()	583
12.278.2.58 fmsubin()	583
12.278.2.59 eq()	583
12.278.2.60 round()	583
12.278.2.61 mod()	583
12.278.2.62 zero()	583
12.278.2.63 sll128()	583
12.278.2.64 srl128()	583
12.278.2.65 vand()	584
12.278.2.66 vor()	584
12.278.2.67 vxor()	584
12.278.2.68 vandnot()	584
12.278.3 Field Documentation	584
12.278.3.1 vect_size	584
12.278.3.2 alignment	584
12.279 Simd128_impl< true, true, false, 8 > Struct Reference	584
12.279.1 Member Typedef Documentation	586
12.279.1.1 scalar_t	586
12.279.1.2 aligned_allocator	586
12.279.1.3 aligned_vector	586
12.279.1.4 is_same_element	587
12.279.1.5 vect_t	587
12.279.2 Member Function Documentation	587
12.279.2.1 type_string()	587
12.279.2.2 set1() [1/2]	587
12.279.2.3 set() [1/2]	587
12.279.2.4 gather() [1/2]	587
12.279.2.5 load() [1/2]	587
12.279.2.6 loadu() [1/2]	587
12.279.2.7 store() [1/2]	587
12.279.2.8 storeu() [1/2]	587
12.279.2.9 stream() [1/2]	587
12.279.2.10 sra()	588
12.279.2.11 greater()	588
12.279.2.12 lesser()	588
12.279.2.13 greater_eq()	588
12.279.2.14 lesser_eq()	588
12.279.2.15 mullo()	588
12.279.2.16 mulhi()	588
12.279.2.17 mulx()	588

12.279.2.18 fmaddx()	588
12.279.2.19 fmaddxin()	588
12.279.2.20 fnmaddx()	589
12.279.2.21 fnmaddxin()	589
12.279.2.22 fmsubx()	589
12.279.2.23 fmsubxin()	589
12.279.2.24 hadd_to_scal()	589
12.279.2.25 valid()	589
12.279.2.26 compliant()	589
12.279.2.27 set1() [2/2]	589
12.279.2.28 set() [2/2]	589
12.279.2.29 gather() [2/2]	590
12.279.2.30 get()	590
12.279.2.31 load() [2/2]	590
12.279.2.32 loadu() [2/2]	590
12.279.2.33 store() [2/2]	590
12.279.2.34 storeu() [2/2]	590
12.279.2.35 stream() [2/2]	590
12.279.2.36 sll()	590
12.279.2.37 srl()	590
12.279.2.38 shuffle()	590
12.279.2.39 unpacklo_intrinsic()	591
12.279.2.40 unpackhi_intrinsic()	591
12.279.2.41 unpacklo()	591
12.279.2.42 unpackhi()	591
12.279.2.43 unpacklohi()	591
12.279.2.44 pack_even()	591
12.279.2.45 pack_odd()	591
12.279.2.46 pack()	591
12.279.2.47 transpose()	591
12.279.2.48 blend()	592
12.279.2.49 add()	592
12.279.2.50 addin()	592
12.279.2.51 sub()	592
12.279.2.52 subin()	592
12.279.2.53 mul()	592
12.279.2.54 fmadd()	592
12.279.2.55 fmaddin()	592
12.279.2.56 fnmadd()	592
12.279.2.57 fnmaddin()	593
12.279.2.58 fmsub()	593
12.279.2.59 fmsubin()	593

12.279.2.60 eq()	593
12.279.2.61 round()	593
12.279.2.62 mask_high()	593
12.279.2.63 mulhi_fast()	593
12.279.2.64 mod()	593
12.279.2.65 signbits()	593
12.279.2.66 zero()	594
12.279.2.67 sll128()	594
12.279.2.68 srl128()	594
12.279.2.69 vand()	594
12.279.2.70 vor()	594
12.279.2.71 vxor()	594
12.279.2.72 vandnot()	594
12.279.3 Field Documentation	594
12.279.3.1 vect_size	594
12.279.3.2 alignment	594
12.280 Simd128_impl< true, true, true, 2 > Struct Reference	594
12.280.1 Member Typedef Documentation	596
12.280.1.1 vect_t	596
12.280.1.2 scalar_t	596
12.280.1.3 aligned_allocator	597
12.280.1.4 aligned_vector	597
12.280.1.5 is_same_element	597
12.280.2 Member Function Documentation	597
12.280.2.1 type_string()	597
12.280.2.2 valid()	597
12.280.2.3 compliant()	597
12.280.2.4 set1()	597
12.280.2.5 set()	597
12.280.2.6 gather()	597
12.280.2.7 load()	597
12.280.2.8 loadu()	598
12.280.2.9 store()	598
12.280.2.10 storeu()	598
12.280.2.11 stream()	598
12.280.2.12 sll()	598
12.280.2.13 srl()	598
12.280.2.14 sra()	598
12.280.2.15 shuffle()	598
12.280.2.16 unpacklo_intrinsic()	598
12.280.2.17 unpackhi_intrinsic()	598
12.280.2.18 unpacklo()	599

12.280.2.19 unpackhi()	599
12.280.2.20 unpacklohi()	599
12.280.2.21 pack_even()	599
12.280.2.22 pack_odd()	599
12.280.2.23 pack()	599
12.280.2.24 transpose()	599
12.280.2.25 blend()	599
12.280.2.26 add()	600
12.280.2.27 addin()	600
12.280.2.28 sub()	600
12.280.2.29 subin()	600
12.280.2.30 mullo()	600
12.280.2.31 mul()	600
12.280.2.32 mulhi()	600
12.280.2.33 mulx()	600
12.280.2.34 fmadd()	600
12.280.2.35 fmaddin()	600
12.280.2.36 fmaddx()	601
12.280.2.37 fmaddxin()	601
12.280.2.38 fnmadd()	601
12.280.2.39 fnmaddin()	601
12.280.2.40 fnmaddx()	601
12.280.2.41 fnmaddxin()	601
12.280.2.42 fmsub()	601
12.280.2.43 fmsubin()	601
12.280.2.44 fmsubx()	601
12.280.2.45 fmsubxin()	602
12.280.2.46 eq()	602
12.280.2.47 greater()	602
12.280.2.48 lesser()	602
12.280.2.49 greater_eq()	602
12.280.2.50 lesser_eq()	602
12.280.2.51 hadd_to_scal()	602
12.280.2.52 round()	602
12.280.2.53 mod()	602
12.280.2.54 zero()	603
12.280.2.55 sl128()	603
12.280.2.56 srl128()	603
12.280.2.57 vand()	603
12.280.2.58 vor()	603
12.280.2.59 vxor()	603
12.280.2.60 vandnot()	603

12.280.3 Field Documentation	603
12.280.3.1 vect_size	603
12.280.3.2 alignment	603
12.281 Simd128_impl< true, true, true, 4 > Struct Reference	604
12.281.1 Member Typedef Documentation	605
12.281.1.1 vect_t	605
12.281.1.2 scalar_t	605
12.281.1.3 aligned_allocator	606
12.281.1.4 aligned_vector	606
12.281.1.5 is_same_element	606
12.281.2 Member Function Documentation	606
12.281.2.1 type_string()	606
12.281.2.2 valid()	606
12.281.2.3 compliant()	606
12.281.2.4 set1()	606
12.281.2.5 set()	606
12.281.2.6 gather()	606
12.281.2.7 load()	606
12.281.2.8 loadu()	606
12.281.2.9 store()	607
12.281.2.10 storeu()	607
12.281.2.11 stream()	607
12.281.2.12 sll()	607
12.281.2.13 srl()	607
12.281.2.14 sra()	607
12.281.2.15 shuffle()	607
12.281.2.16 unpacklo_intrinsic()	607
12.281.2.17 unpackhi_intrinsic()	607
12.281.2.18 unpacklo()	607
12.281.2.19 unpackhi()	608
12.281.2.20 unpacklohi()	608
12.281.2.21 pack_even()	608
12.281.2.22 pack_odd()	608
12.281.2.23 pack()	608
12.281.2.24 transpose()	608
12.281.2.25 blend()	608
12.281.2.26 add()	608
12.281.2.27 addin()	608
12.281.2.28 sub()	609
12.281.2.29 subin()	609
12.281.2.30 mullo()	609
12.281.2.31 mul()	609

12.281.2.32 mulhi()	609
12.281.2.33 mulx()	609
12.281.2.34 fmadd()	609
12.281.2.35 fmaddin()	609
12.281.2.36 fmaddx()	609
12.281.2.37 fmaddxin()	610
12.281.2.38 fnmadd()	610
12.281.2.39 fnmaddin()	610
12.281.2.40 fnmaddx()	610
12.281.2.41 fnmaddxin()	610
12.281.2.42 fmsub()	610
12.281.2.43 fmsubin()	610
12.281.2.44 fmsubx()	610
12.281.2.45 fmsubxin()	610
12.281.2.46 eq()	611
12.281.2.47 greater()	611
12.281.2.48 lesser()	611
12.281.2.49 greater_eq()	611
12.281.2.50 lesser_eq()	611
12.281.2.51 hadd_to_scal()	611
12.281.2.52 round()	611
12.281.2.53 mod()	611
12.281.2.54 zero()	611
12.281.2.55 sll128()	612
12.281.2.56 srl128()	612
12.281.2.57 vand()	612
12.281.2.58 vor()	612
12.281.2.59 vxor()	612
12.281.2.60 vandnot()	612
12.281.3 Field Documentation	612
12.281.3.1 vect_size	612
12.281.3.2 alignment	612
12.282 Simd128_impl< true, true, true, 8 > Struct Reference	612
12.282.1 Member Typedef Documentation	614
12.282.1.1 vect_t	614
12.282.1.2 scalar_t	615
12.282.1.3 aligned_allocator	615
12.282.1.4 aligned_vector	615
12.282.1.5 is_same_element	615
12.282.2 Member Function Documentation	615
12.282.2.1 type_string()	615
12.282.2.2 valid()	615

12.282.2.3 compliant()	615
12.282.2.4 set1()	615
12.282.2.5 set()	615
12.282.2.6 gather()	615
12.282.2.7 get()	615
12.282.2.8 load()	616
12.282.2.9 loadu()	616
12.282.2.10 store()	616
12.282.2.11 storeu()	616
12.282.2.12 stream()	616
12.282.2.13 sll()	616
12.282.2.14 srl()	616
12.282.2.15 sra()	616
12.282.2.16 shuffle()	616
12.282.2.17 unpacklo_intrinsic()	616
12.282.2.18 unpackhi_intrinsic()	617
12.282.2.19 unpacklo()	617
12.282.2.20 unpackhi()	617
12.282.2.21 unpacklohi()	617
12.282.2.22 pack_even()	617
12.282.2.23 pack_odd()	617
12.282.2.24 pack()	617
12.282.2.25 transpose()	617
12.282.2.26 blend()	617
12.282.2.27 add()	618
12.282.2.28 addin()	618
12.282.2.29 sub()	618
12.282.2.30 subin()	618
12.282.2.31 mullo()	618
12.282.2.32 mul()	618
12.282.2.33 mulhi()	618
12.282.2.34 mulx()	618
12.282.2.35 fmadd()	618
12.282.2.36 fmaddin()	618
12.282.2.37 fmaddx()	619
12.282.2.38 fmaddxin()	619
12.282.2.39 fnmadd()	619
12.282.2.40 fnmaddin()	619
12.282.2.41 fnmaddx()	619
12.282.2.42 fnmaddxin()	619
12.282.2.43 fmsub()	619
12.282.2.44 fmsubin()	619



12.282.2.45 fmsubx()	619
12.282.2.46 fmsubxin()	620
12.282.2.47 eq()	620
12.282.2.48 greater()	620
12.282.2.49 lesser()	620
12.282.2.50 greater_eq()	620
12.282.2.51 lesser_eq()	620
12.282.2.52 hadd_to_scal()	620
12.282.2.53 round()	620
12.282.2.54 mask_high()	620
12.282.2.55 mulhi_fast()	620
12.282.2.56 mod()	621
12.282.2.57 signbits()	621
12.282.2.58 zero()	621
12.282.2.59 sll128()	621
12.282.2.60 srl128()	621
12.282.2.61 vand()	621
12.282.2.62 vor()	621
12.282.2.63 vxor()	621
12.282.2.64 vandnot()	621
12.282.3 Field Documentation	622
12.282.3.1 vect_size	622
12.282.3.2 alignment	622
12.283 Simd128i_base Struct Reference	622
12.283.1 Member Typedef Documentation	622
12.283.1.1 vect_t	622
12.283.2 Member Function Documentation	622
12.283.2.1 zero()	622
12.283.2.2 sll128()	622
12.283.2.3 srl128()	623
12.283.2.4 vand()	623
12.283.2.5 vor()	623
12.283.2.6 vxor()	623
12.283.2.7 vandnot()	623
12.284 Simd256_impl< ArithType, Int, Signed, Size > Struct Template Reference	623
12.285 Simd256_impl< true, false, true, 4 > Struct Reference	623
12.286 Simd256_impl< true, false, true, 8 > Struct Reference	624
12.286.1 Member Typedef Documentation	625
12.286.1.1 vect_t	625
12.286.1.2 scalar_t	625
12.286.1.3 aligned_allocator	625
12.286.1.4 aligned_vector	625

12.286.1.5 is_same_element . . . . .	625
12.286.2 Member Function Documentation . . . . .	625
12.286.2.1 type_string() . . . . .	625
12.286.2.2 valid() . . . . .	626
12.286.2.3 compliant() . . . . .	626
12.286.2.4 zero() . . . . .	626
12.286.2.5 set1() . . . . .	626
12.286.2.6 set() . . . . .	626
12.286.2.7 gather() . . . . .	626
12.286.2.8 load() . . . . .	626
12.286.2.9 loadu() . . . . .	626
12.286.2.10 store() . . . . .	626
12.286.2.11 storeu() . . . . .	626
12.286.2.12 stream() . . . . .	627
12.286.2.13 unpacklo_intrinsic() . . . . .	627
12.286.2.14 unpackhi_intrinsic() . . . . .	627
12.286.2.15 unpacklo() . . . . .	627
12.286.2.16 unpackhi() . . . . .	627
12.286.2.17 unpacklohi() . . . . .	627
12.286.2.18 pack_even() . . . . .	627
12.286.2.19 pack_odd() . . . . .	627
12.286.2.20 pack() . . . . .	627
12.286.2.21 transpose() . . . . .	628
12.286.2.22 blend() . . . . .	628
12.286.2.23 blendv() . . . . .	628
12.286.2.24 add() . . . . .	628
12.286.2.25 addin() . . . . .	628
12.286.2.26 sub() . . . . .	628
12.286.2.27 subin() . . . . .	628
12.286.2.28 mul() . . . . .	628
12.286.2.29 mulin() . . . . .	628
12.286.2.30 div() . . . . .	629
12.286.2.31 fmadd() . . . . .	629
12.286.2.32 fmaddin() . . . . .	629
12.286.2.33 fnmadd() . . . . .	629
12.286.2.34 fnmaddin() . . . . .	629
12.286.2.35 fmsub() . . . . .	629
12.286.2.36 fmsubin() . . . . .	629
12.286.2.37 eq() . . . . .	629
12.286.2.38 lesser() . . . . .	629
12.286.2.39 lesser_eq() . . . . .	630
12.286.2.40 greater() . . . . .	630

12.286.2.41 greater_eq()	630
12.286.2.42 vand()	630
12.286.2.43 vor()	630
12.286.2.44 vxor()	630
12.286.2.45 vandnot()	630
12.286.2.46 floor()	630
12.286.2.47 ceil()	630
12.286.2.48 round()	630
12.286.2.49 hadd()	631
12.286.2.50 hadd_to_scal()	631
12.286.2.51 mod()	631
12.286.3 Field Documentation	631
12.286.3.1 vect_size	631
12.286.3.2 alignment	631
12.287 Simd256_impl< true, true, false, 2 > Struct Reference	631
12.287.1 Member Typedef Documentation	633
12.287.1.1 scalar_t	633
12.287.1.2 aligned_allocator	633
12.287.1.3 aligned_vector	633
12.287.1.4 is_same_element	633
12.287.1.5 simdHalf	634
12.287.1.6 vect_t	634
12.287.1.7 half_t	634
12.287.2 Member Function Documentation	634
12.287.2.1 type_string()	634
12.287.2.2 set1() [1/2]	634
12.287.2.3 set() [1/2]	634
12.287.2.4 gather() [1/2]	634
12.287.2.5 load() [1/2]	634
12.287.2.6 loadu() [1/2]	634
12.287.2.7 store() [1/2]	635
12.287.2.8 storeu() [1/2]	635
12.287.2.9 stream() [1/2]	635
12.287.2.10 sra()	635
12.287.2.11 greater()	635
12.287.2.12 lesser()	635
12.287.2.13 greater_eq()	635
12.287.2.14 lesser_eq()	635
12.287.2.15 mulhi()	635
12.287.2.16 mulx()	635
12.287.2.17 fmaddx()	636
12.287.2.18 fmaddxin()	636

12.287.2.19 fnmaddx()	636
12.287.2.20 fnmaddxin()	636
12.287.2.21 fmsubx()	636
12.287.2.22 fmsubxin()	636
12.287.2.23 hadd_to_scal()	636
12.287.2.24 valid()	636
12.287.2.25 compliant()	636
12.287.2.26 set1() [2/2]	637
12.287.2.27 set() [2/2]	637
12.287.2.28 gather() [2/2]	637
12.287.2.29 load() [2/2]	637
12.287.2.30 loadu() [2/2]	637
12.287.2.31 store() [2/2]	637
12.287.2.32 storeu() [2/2]	637
12.287.2.33 stream() [2/2]	637
12.287.2.34 sll()	638
12.287.2.35 srl()	638
12.287.2.36 shuffle()	638
12.287.2.37 unpacklo_intrinsic()	638
12.287.2.38 unpackhi_intrinsic()	638
12.287.2.39 unpacklo()	638
12.287.2.40 unpackhi()	638
12.287.2.41 unpacklohi()	638
12.287.2.42 pack_even()	638
12.287.2.43 pack_odd()	638
12.287.2.44 pack()	639
12.287.2.45 transpose()	639
12.287.2.46 blend() [1/2]	639
12.287.2.47 blend() [2/2]	639
12.287.2.48 add()	639
12.287.2.49 addin()	639
12.287.2.50 sub()	639
12.287.2.51 subin()	640
12.287.2.52 mullo()	640
12.287.2.53 mul()	640
12.287.2.54 fmadd()	640
12.287.2.55 fmaddin()	640
12.287.2.56 fnmadd()	640
12.287.2.57 fnmaddin()	640
12.287.2.58 fmsub()	640
12.287.2.59 fmsubin()	640
12.287.2.60 eq()	641

12.287.2.61 round()	641
12.287.2.62 mod()	641
12.287.2.63 zero()	641
12.287.3 Field Documentation	641
12.287.3.1 vect_size	641
12.287.3.2 alignment	641
12.288 Simd256_impl< true, true, false, 4 > Struct Reference	641
12.288.1 Member Typedef Documentation	645
12.288.1.1 scalar_t	645
12.288.1.2 aligned_allocator	645
12.288.1.3 aligned_vector	645
12.288.1.4 is_same_element	645
12.288.1.5 simdHalf	645
12.288.1.6 vect_t [1/2]	645
12.288.1.7 vect_t [2/2]	645
12.288.1.8 half_t [1/2]	645
12.288.1.9 half_t [2/2]	645
12.288.2 Member Function Documentation	645
12.288.2.1 type_string() [1/2]	645
12.288.2.2 set1() [1/3]	645
12.288.2.3 set() [1/4]	646
12.288.2.4 gather() [1/3]	646
12.288.2.5 load() [1/3]	646
12.288.2.6 loadu() [1/3]	646
12.288.2.7 store() [1/3]	646
12.288.2.8 storeu() [1/3]	646
12.288.2.9 stream() [1/3]	646
12.288.2.10 sra() [1/2]	646
12.288.2.11 greater() [1/2]	646
12.288.2.12 lesser() [1/2]	647
12.288.2.13 greater_eq() [1/2]	647
12.288.2.14 lesser_eq() [1/2]	647
12.288.2.15 mulhi() [1/2]	647
12.288.2.16 mulx() [1/2]	647
12.288.2.17 fmaddx() [1/2]	647
12.288.2.18 fmaddxin() [1/2]	647
12.288.2.19 fnmaddx() [1/2]	647
12.288.2.20 fnmaddxin() [1/2]	647
12.288.2.21 fmsubx() [1/2]	648
12.288.2.22 fmsubxin() [1/2]	648
12.288.2.23 hadd_to_scal() [1/2]	648
12.288.2.24 type_string() [2/2]	648

12.288.2.25 set1() [2/3]	648
12.288.2.26 set() [2/4]	648
12.288.2.27 gather() [2/3]	648
12.288.2.28 load() [2/3]	648
12.288.2.29 loadu() [2/3]	648
12.288.2.30 store() [2/3]	649
12.288.2.31 storeu() [2/3]	649
12.288.2.32 stream() [2/3]	649
12.288.2.33 sra() [2/2]	649
12.288.2.34 greater() [2/2]	649
12.288.2.35 lesser() [2/2]	649
12.288.2.36 greater_eq() [2/2]	649
12.288.2.37 lesser_eq() [2/2]	649
12.288.2.38 mulhi() [2/2]	649
12.288.2.39 mulx() [2/2]	649
12.288.2.40 fmaddx() [2/2]	650
12.288.2.41 fmaddxin() [2/2]	650
12.288.2.42 fnmaddx() [2/2]	650
12.288.2.43 fnmaddxin() [2/2]	650
12.288.2.44 fmsubx() [2/2]	650
12.288.2.45 fmsubxin() [2/2]	650
12.288.2.46 hadd_to_scal() [2/2]	650
12.288.2.47 is_same_element()	650
12.288.2.48 valid() [1/2]	650
12.288.2.49 valid() [2/2]	651
12.288.2.50 compliant() [1/2]	651
12.288.2.51 compliant() [2/2]	651
12.288.2.52 set1() [3/3]	651
12.288.2.53 set() [3/4]	651
12.288.2.54 set() [4/4]	651
12.288.2.55 gather() [3/3]	651
12.288.2.56 load() [3/3]	652
12.288.2.57 loadu() [3/3]	652
12.288.2.58 store() [3/3]	652
12.288.2.59 storeu() [3/3]	652
12.288.2.60 stream() [3/3]	652
12.288.2.61 sll() [1/2]	652
12.288.2.62 sll() [2/2]	652
12.288.2.63 srl() [1/2]	652
12.288.2.64 srl() [2/2]	652
12.288.2.65 shuffle_twice() [1/2]	652
12.288.2.66 shuffle_twice() [2/2]	653

12.288.2.67 shuffle() [1/2]	653
12.288.2.68 shuffle() [2/2]	653
12.288.2.69 unpacklo_intrinsic() [1/2]	653
12.288.2.70 unpacklo_intrinsic() [2/2]	653
12.288.2.71 unpackhi_intrinsic() [1/2]	653
12.288.2.72 unpackhi_intrinsic() [2/2]	653
12.288.2.73 unpacklo() [1/2]	653
12.288.2.74 unpacklo() [2/2]	653
12.288.2.75 unpackhi() [1/2]	653
12.288.2.76 unpackhi() [2/2]	654
12.288.2.77 unpacklohi() [1/2]	654
12.288.2.78 unpacklohi() [2/2]	654
12.288.2.79 pack_even() [1/2]	654
12.288.2.80 pack_even() [2/2]	654
12.288.2.81 pack_odd() [1/2]	654
12.288.2.82 pack_odd() [2/2]	654
12.288.2.83 pack() [1/2]	654
12.288.2.84 pack() [2/2]	654
12.288.2.85 transpose() [1/2]	655
12.288.2.86 transpose() [2/2]	655
12.288.2.87 blend() [1/2]	655
12.288.2.88 blend() [2/2]	655
12.288.2.89 add() [1/2]	655
12.288.2.90 add() [2/2]	655
12.288.2.91 addin() [1/2]	656
12.288.2.92 addin() [2/2]	656
12.288.2.93 sub() [1/2]	656
12.288.2.94 sub() [2/2]	656
12.288.2.95 subin() [1/2]	656
12.288.2.96 subin() [2/2]	656
12.288.2.97 mullo() [1/2]	656
12.288.2.98 mullo() [2/2]	656
12.288.2.99 mul() [1/2]	656
12.288.2.100 mul() [2/2]	656
12.288.2.101 fmadd() [1/2]	657
12.288.2.102 fmadd() [2/2]	657
12.288.2.103 fmaddin() [1/2]	657
12.288.2.104 fmaddin() [2/2]	657
12.288.2.105 fnmadd() [1/2]	657
12.288.2.106 fnmadd() [2/2]	657
12.288.2.107 fnmaddin() [1/2]	657
12.288.2.108 fnmaddin() [2/2]	657

12.288.2.109 fmsub() [1/2]	657
12.288.2.110 fmsub() [2/2]	658
12.288.2.111 fmsubin() [1/2]	658
12.288.2.112 fmsubin() [2/2]	658
12.288.2.113 eq() [1/2]	658
12.288.2.114 eq() [2/2]	658
12.288.2.115 round() [1/2]	658
12.288.2.116 round() [2/2]	658
12.288.2.117 mod() [1/2]	658
12.288.2.118 mod() [2/2]	659
12.288.2.119 zero() [1/2]	659
12.288.2.120 zero() [2/2]	659
12.288.2.121 vor()	659
12.288.2.122 vxor()	659
12.288.2.123 vand()	659
12.288.2.124 vandnot()	659
12.288.3 Field Documentation	659
12.288.3.1 vect_size	659
12.288.3.2 alignment	659
12.289 Simd256_impl< true, true, false, 8 > Struct Reference	660
12.289.1 Member Typedef Documentation	662
12.289.1.1 scalar_t	662
12.289.1.2 aligned_allocator	662
12.289.1.3 aligned_vector	662
12.289.1.4 is_same_element	662
12.289.1.5 simdHalf	662
12.289.1.6 vect_t	662
12.289.1.7 half_t	662
12.289.2 Member Function Documentation	662
12.289.2.1 type_string()	662
12.289.2.2 set1() [1/2]	662
12.289.2.3 set() [1/2]	662
12.289.2.4 gather() [1/2]	662
12.289.2.5 load() [1/2]	663
12.289.2.6 loadu() [1/2]	663
12.289.2.7 store() [1/2]	663
12.289.2.8 storeu() [1/2]	663
12.289.2.9 stream() [1/2]	663
12.289.2.10 sra()	663
12.289.2.11 greater()	663
12.289.2.12 lesser()	663
12.289.2.13 greater_eq()	663



12.289.2.14 lesser_eq()	663
12.289.2.15 mullo()	664
12.289.2.16 mulhi()	664
12.289.2.17 mulx()	664
12.289.2.18 fmaddx()	664
12.289.2.19 fmaddxin()	664
12.289.2.20 fnmaddx()	664
12.289.2.21 fnmaddxin()	664
12.289.2.22 fmsubx()	664
12.289.2.23 fmsubxin()	664
12.289.2.24 hadd_to_scal()	665
12.289.2.25 valid()	665
12.289.2.26 compliant()	665
12.289.2.27 set1() [2/2]	665
12.289.2.28 set() [2/2]	665
12.289.2.29 gather() [2/2]	665
12.289.2.30 get()	665
12.289.2.31 load() [2/2]	665
12.289.2.32 loadu() [2/2]	665
12.289.2.33 store() [2/2]	665
12.289.2.34 storeu() [2/2]	666
12.289.2.35 stream() [2/2]	666
12.289.2.36 sll()	666
12.289.2.37 srl()	666
12.289.2.38 shuffle()	666
12.289.2.39 unpacklo_intrinsic()	666
12.289.2.40 unpackhi_intrinsic()	666
12.289.2.41 unpacklo()	666
12.289.2.42 unpackhi()	666
12.289.2.43 unpacklohi()	666
12.289.2.44 pack_even()	667
12.289.2.45 pack_odd()	667
12.289.2.46 pack()	667
12.289.2.47 transpose()	667
12.289.2.48 blend()	667
12.289.2.49 add()	667
12.289.2.50 addin()	667
12.289.2.51 sub()	667
12.289.2.52 subin()	667
12.289.2.53 mul()	668
12.289.2.54 fmadd()	668
12.289.2.55 fmaddin()	668

12.289.2.56 fnmadd()	668
12.289.2.57 fnmaddin()	668
12.289.2.58 fmsub()	668
12.289.2.59 fmsubin()	668
12.289.2.60 eq()	668
12.289.2.61 round()	668
12.289.2.62 mask_high()	669
12.289.2.63 mulhi_fast()	669
12.289.2.64 mod()	669
12.289.2.65 signbits()	669
12.289.2.66 zero()	669
12.289.3 Field Documentation	669
12.289.3.1 vect_size	669
12.289.3.2 alignment	669
12.290 Simd256_impl< true, true, true, 2 > Struct Reference	669
12.290.1 Member Typedef Documentation	671
12.290.1.1 vect_t	671
12.290.1.2 half_t	671
12.290.1.3 scalar_t	671
12.290.1.4 simdHalf	671
12.290.1.5 aligned_allocator	671
12.290.1.6 aligned_vector	672
12.290.1.7 is_same_element	672
12.290.2 Member Function Documentation	672
12.290.2.1 type_string()	672
12.290.2.2 valid()	672
12.290.2.3 compliant()	672
12.290.2.4 set1()	672
12.290.2.5 set()	672
12.290.2.6 gather()	672
12.290.2.7 load()	673
12.290.2.8 loadu()	673
12.290.2.9 store()	673
12.290.2.10 storeu()	673
12.290.2.11 stream()	673
12.290.2.12 sll()	673
12.290.2.13 srl()	673
12.290.2.14 sra()	673
12.290.2.15 shuffle()	673
12.290.2.16 unpacklo_intrinsic()	673
12.290.2.17 unpackhi_intrinsic()	674
12.290.2.18 unpacklo()	674

12.290.2.19 unpackhi()	674
12.290.2.20 unpacklohi()	674
12.290.2.21 pack_even()	674
12.290.2.22 pack_odd()	674
12.290.2.23 pack()	674
12.290.2.24 transpose()	674
12.290.2.25 blend() [1/2]	675
12.290.2.26 blend() [2/2]	675
12.290.2.27 add()	675
12.290.2.28 addin()	675
12.290.2.29 sub()	675
12.290.2.30 subin()	675
12.290.2.31 mullo()	675
12.290.2.32 mul()	675
12.290.2.33 mulhi()	676
12.290.2.34 mulx()	676
12.290.2.35 fmadd()	676
12.290.2.36 fmaddin()	676
12.290.2.37 fmaddx()	676
12.290.2.38 fmaddxin()	676
12.290.2.39 fnmadd()	676
12.290.2.40 fnmaddin()	676
12.290.2.41 fnmaddx()	676
12.290.2.42 fnmaddxin()	677
12.290.2.43 fmsub()	677
12.290.2.44 fmsubin()	677
12.290.2.45 fmsubx()	677
12.290.2.46 fmsubxin()	677
12.290.2.47 eq()	677
12.290.2.48 greater()	677
12.290.2.49 lesser()	677
12.290.2.50 greater_eq()	677
12.290.2.51 lesser_eq()	678
12.290.2.52 hadd_to_scal()	678
12.290.2.53 round()	678
12.290.2.54 mod()	678
12.290.2.55 zero()	678
12.290.3 Field Documentation	678
12.290.3.1 vect_size	678
12.290.3.2 alignment	678
12.291 Simd256_impl< true, true, true, 4 > Struct Reference	678
12.291.1 Member Typedef Documentation	682

12.291.1.1 vect_t [1/2]	682
12.291.1.2 half_t [1/2]	682
12.291.1.3 scalar_t	682
12.291.1.4 simdHalf [1/2]	682
12.291.1.5 aligned_allocator	682
12.291.1.6 aligned_vector	682
12.291.1.7 is_same_element	682
12.291.1.8 vect_t [2/2]	682
12.291.1.9 half_t [2/2]	682
12.291.1.10 simdHalf [2/2]	682
12.291.2 Member Function Documentation	682
12.291.2.1 type_string() [1/2]	682
12.291.2.2 valid() [1/2]	682
12.291.2.3 compliant() [1/2]	683
12.291.2.4 set1() [1/2]	683
12.291.2.5 set() [1/2]	683
12.291.2.6 gather() [1/2]	683
12.291.2.7 load() [1/2]	683
12.291.2.8 loadu() [1/2]	683
12.291.2.9 store() [1/2]	683
12.291.2.10 storeu() [1/2]	683
12.291.2.11 stream() [1/2]	683
12.291.2.12 sll() [1/2]	684
12.291.2.13 srl() [1/2]	684
12.291.2.14 sra() [1/2]	684
12.291.2.15 shuffle_twice() [1/2]	684
12.291.2.16 shuffle() [1/2]	684
12.291.2.17 unpacklo_intrinsic() [1/2]	684
12.291.2.18 unpackhi_intrinsic() [1/2]	684
12.291.2.19 unpacklo() [1/2]	684
12.291.2.20 unpackhi() [1/2]	684
12.291.2.21 unpacklohi() [1/2]	684
12.291.2.22 pack_even() [1/2]	685
12.291.2.23 pack_odd() [1/2]	685
12.291.2.24 pack() [1/2]	685
12.291.2.25 transpose() [1/2]	685
12.291.2.26 blend() [1/2]	685
12.291.2.27 add() [1/2]	685
12.291.2.28 addin() [1/2]	685
12.291.2.29 sub() [1/2]	685
12.291.2.30 subin() [1/2]	686
12.291.2.31 mullo() [1/2]	686

12.291.2.32 mul() [1/2]	686
12.291.2.33 mulhi() [1/2]	686
12.291.2.34 mulx() [1/2]	686
12.291.2.35 fmadd() [1/2]	686
12.291.2.36 fmaddin() [1/2]	686
12.291.2.37 fmaddx() [1/2]	686
12.291.2.38 fmaddxin() [1/2]	686
12.291.2.39 fnmadd() [1/2]	687
12.291.2.40 fnmaddin() [1/2]	687
12.291.2.41 fnmaddx() [1/2]	687
12.291.2.42 fnmaddxin() [1/2]	687
12.291.2.43 fmsub() [1/2]	687
12.291.2.44 fmsubin() [1/2]	687
12.291.2.45 fmsubx() [1/2]	687
12.291.2.46 fmsubxin() [1/2]	687
12.291.2.47 eq() [1/2]	688
12.291.2.48 greater() [1/2]	688
12.291.2.49 lesser() [1/2]	688
12.291.2.50 greater_eq() [1/2]	688
12.291.2.51 lesser_eq() [1/2]	688
12.291.2.52 hadd_to_scal() [1/2]	688
12.291.2.53 round() [1/2]	688
12.291.2.54 mod() [1/2]	688
12.291.2.55 type_string() [2/2]	688
12.291.2.56 valid() [2/2]	688
12.291.2.57 compliant() [2/2]	689
12.291.2.58 set1() [2/2]	689
12.291.2.59 set() [2/2]	689
12.291.2.60 gather() [2/2]	689
12.291.2.61 load() [2/2]	689
12.291.2.62 loadu() [2/2]	689
12.291.2.63 store() [2/2]	689
12.291.2.64 storeu() [2/2]	689
12.291.2.65 stream() [2/2]	690
12.291.2.66 sll() [2/2]	690
12.291.2.67 srl() [2/2]	690
12.291.2.68 sra() [2/2]	690
12.291.2.69 shuffle_twice() [2/2]	690
12.291.2.70 shuffle() [2/2]	690
12.291.2.71 unpacklo_intrinsic() [2/2]	690
12.291.2.72 unpackhi_intrinsic() [2/2]	690
12.291.2.73 unpacklo() [2/2]	690

12.291.2.74 unpackhi() [2/2]	690
12.291.2.75 unpacklohi() [2/2]	691
12.291.2.76 pack_even() [2/2]	691
12.291.2.77 pack_odd() [2/2]	691
12.291.2.78 pack() [2/2]	691
12.291.2.79 transpose() [2/2]	691
12.291.2.80 blend() [2/2]	691
12.291.2.81 add() [2/2]	691
12.291.2.82 addin() [2/2]	692
12.291.2.83 sub() [2/2]	692
12.291.2.84 subin() [2/2]	692
12.291.2.85 mullo() [2/2]	692
12.291.2.86 mul() [2/2]	692
12.291.2.87 mulhi() [2/2]	692
12.291.2.88 mulx() [2/2]	692
12.291.2.89 fmadd() [2/2]	692
12.291.2.90 fmaddin() [2/2]	692
12.291.2.91 fmaddx() [2/2]	693
12.291.2.92 fmaddxin() [2/2]	693
12.291.2.93 fnmadd() [2/2]	693
12.291.2.94 fnmaddin() [2/2]	693
12.291.2.95 fnmaddx() [2/2]	693
12.291.2.96 fnmaddxin() [2/2]	693
12.291.2.97 fmsub() [2/2]	693
12.291.2.98 fmsubin() [2/2]	693
12.291.2.99 fmsubx() [2/2]	693
12.291.2.100 fmsubxin() [2/2]	694
12.291.2.101 eq() [2/2]	694
12.291.2.102 greater() [2/2]	694
12.291.2.103 lesser() [2/2]	694
12.291.2.104 greater_eq() [2/2]	694
12.291.2.105 lesser_eq() [2/2]	694
12.291.2.106 hadd_to_scal() [2/2]	694
12.291.2.107 round() [2/2]	694
12.291.2.108 mod() [2/2]	694
12.291.2.109 is_same_element()	695
12.291.2.110 zero() [1/2]	695
12.291.2.111 zero() [2/2]	695
12.291.2.112 vor()	695
12.291.2.113 vxor()	695
12.291.2.114 vand()	695
12.291.2.115 vandnot()	695

12.291.3 Field Documentation	695
12.291.3.1 vect_size	695
12.291.3.2 alignment	695
12.292 Simd256_impl< true, true, true, 8 > Struct Reference	695
12.292.1 Member Typedef Documentation	697
12.292.1.1 vect_t	697
12.292.1.2 half_t	697
12.292.1.3 scalar_t	698
12.292.1.4 simdHalf	698
12.292.1.5 aligned_allocator	698
12.292.1.6 aligned_vector	698
12.292.1.7 is_same_element	698
12.292.2 Member Function Documentation	698
12.292.2.1 type_string()	698
12.292.2.2 valid()	698
12.292.2.3 compliant()	698
12.292.2.4 set1()	698
12.292.2.5 set()	698
12.292.2.6 gather()	698
12.292.2.7 get()	699
12.292.2.8 load()	699
12.292.2.9 loadu()	699
12.292.2.10 store()	699
12.292.2.11 storeu()	699
12.292.2.12 stream()	699
12.292.2.13 sll()	699
12.292.2.14 srl()	699
12.292.2.15 sra()	699
12.292.2.16 shuffle()	699
12.292.2.17 unpacklo_intrinsic()	700
12.292.2.18 unpackhi_intrinsic()	700
12.292.2.19 unpacklo()	700
12.292.2.20 unpackhi()	700
12.292.2.21 unpacklohi()	700
12.292.2.22 pack_even()	700
12.292.2.23 pack_odd()	700
12.292.2.24 pack()	700
12.292.2.25 transpose()	700
12.292.2.26 blend()	701
12.292.2.27 add()	701
12.292.2.28 addin()	701
12.292.2.29 sub()	701

12.292.2.30 subin()	701
12.292.2.31 mullo()	701
12.292.2.32 mul()	701
12.292.2.33 mulhi()	701
12.292.2.34 mulx()	701
12.292.2.35 fmadd()	701
12.292.2.36 fmaddin()	702
12.292.2.37 fmaddx()	702
12.292.2.38 fmaddxin()	702
12.292.2.39 fnmadd()	702
12.292.2.40 fnmaddin()	702
12.292.2.41 fnmaddx()	702
12.292.2.42 fnmaddxin()	702
12.292.2.43 fmsub()	702
12.292.2.44 fmsubin()	702
12.292.2.45 fmsubx()	703
12.292.2.46 fmsubxin()	703
12.292.2.47 eq()	703
12.292.2.48 greater()	703
12.292.2.49 lesser()	703
12.292.2.50 greater_eq()	703
12.292.2.51 lesser_eq()	703
12.292.2.52 hadd_to_scal()	703
12.292.2.53 round()	703
12.292.2.54 mask_high()	703
12.292.2.55 mulhi_fast()	704
12.292.2.56 mod()	704
12.292.2.57 signbits()	704
12.292.2.58 zero()	704
12.292.3 Field Documentation	704
12.292.3.1 vect_size	704
12.292.3.2 alignment	704
12.293 Simd256fp_base Struct Reference	704
12.294 Simd256i_base Struct Reference	705
12.294.1 Member Typedef Documentation	705
12.294.1.1 vect_t	705
12.294.2 Member Function Documentation	705
12.294.2.1 zero()	705
12.295 Simd512_impl< ArithType, Int, Signed, Size > Struct Template Reference	705
12.296 Simd512_impl< true, false, true, 4 > Struct Reference	705
12.297 Simd512_impl< true, false, true, 8 > Struct Reference	705
12.297.1 Member Typedef Documentation	707



12.297.1.1 vect_t	707
12.297.1.2 scalar_t	707
12.297.1.3 aligned_allocator	707
12.297.1.4 aligned_vector	707
12.297.1.5 is_same_element	707
12.297.2 Member Function Documentation	707
12.297.2.1 type_string()	707
12.297.2.2 valid()	707
12.297.2.3 compliant()	707
12.297.2.4 zero()	707
12.297.2.5 set1()	707
12.297.2.6 set()	708
12.297.2.7 gather()	708
12.297.2.8 load()	708
12.297.2.9 loadu()	708
12.297.2.10 store()	708
12.297.2.11 storeu()	708
12.297.2.12 stream()	708
12.297.2.13 shuffle()	708
12.297.2.14 unpacklo_intrinsic()	708
12.297.2.15 unpackhi_intrinsic()	709
12.297.2.16 unpacklo()	709
12.297.2.17 unpackhi()	709
12.297.2.18 unpacklohi()	709
12.297.2.19 pack_even()	709
12.297.2.20 pack_odd()	709
12.297.2.21 pack()	709
12.297.2.22 transpose()	709
12.297.2.23 blend()	710
12.297.2.24 blendv()	710
12.297.2.25 add()	710
12.297.2.26 addin()	710
12.297.2.27 sub()	710
12.297.2.28 subin()	710
12.297.2.29 mul()	710
12.297.2.30 mulin()	710
12.297.2.31 div()	710
12.297.2.32 fmadd()	711
12.297.2.33 fmaddin()	711
12.297.2.34 fnmadd()	711
12.297.2.35 fnmaddin()	711
12.297.2.36 fmsub()	711

12.297.2.37 fmsubin()	711
12.297.2.38 eq()	711
12.297.2.39 lesser()	711
12.297.2.40 lesser_eq()	711
12.297.2.41 greater()	712
12.297.2.42 greater_eq()	712
12.297.2.43 floor()	712
12.297.2.44 ceil()	712
12.297.2.45 round()	712
12.297.2.46 hadd()	712
12.297.2.47 hadd_to_scal()	712
12.297.3 Field Documentation	712
12.297.3.1 vect_size	712
12.297.3.2 alignment	712
12.298 Simd512_impl< true, true, false, 8 > Struct Reference	712
12.298.1 Member Typedef Documentation	715
12.298.1.1 scalar_t	715
12.298.1.2 aligned_allocator	715
12.298.1.3 aligned_vector	715
12.298.1.4 is_same_element	715
12.298.1.5 simdHalf	715
12.298.1.6 vect_t	715
12.298.1.7 half_t	715
12.298.2 Member Function Documentation	715
12.298.2.1 type_string()	715
12.298.2.2 set1() [1/2]	715
12.298.2.3 set() [1/3]	715
12.298.2.4 gather() [1/2]	716
12.298.2.5 load() [1/2]	716
12.298.2.6 loadu() [1/2]	716
12.298.2.7 store() [1/2]	716
12.298.2.8 maskstore() [1/2]	716
12.298.2.9 storeu() [1/2]	716
12.298.2.10 stream() [1/2]	716
12.298.2.11 sra()	716
12.298.2.12 greater()	716
12.298.2.13 lesser()	716
12.298.2.14 greater_eq()	717
12.298.2.15 lesser_eq()	717
12.298.2.16 mullo()	717
12.298.2.17 mulhi()	717
12.298.2.18 mulx()	717

12.298.2.19 fmaddx()	717
12.298.2.20 fmaddxin()	717
12.298.2.21 fnmaddx()	717
12.298.2.22 fnmaddxin()	717
12.298.2.23 fmsubx()	718
12.298.2.24 fmsubxin()	718
12.298.2.25 hadd_to_scal()	718
12.298.2.26 valid()	718
12.298.2.27 compliant()	718
12.298.2.28 set1() [2/2]	718
12.298.2.29 set() [2/3]	718
12.298.2.30 set() [3/3]	718
12.298.2.31 gather() [2/2]	718
12.298.2.32 load() [2/2]	719
12.298.2.33 loadu() [2/2]	719
12.298.2.34 store() [2/2]	719
12.298.2.35 maskstore() [2/2]	719
12.298.2.36 storeu() [2/2]	719
12.298.2.37 stream() [2/2]	719
12.298.2.38 sll()	719
12.298.2.39 srl()	719
12.298.2.40 shuffle()	719
12.298.2.41 unpacklo_intrinsic()	720
12.298.2.42 unpackhi_intrinsic()	720
12.298.2.43 unpacklo()	720
12.298.2.44 unpackhi()	720
12.298.2.45 unpacklohi()	720
12.298.2.46 pack_even()	720
12.298.2.47 pack_odd()	720
12.298.2.48 pack()	720
12.298.2.49 transpose()	720
12.298.2.50 blend()	721
12.298.2.51 add()	721
12.298.2.52 addin()	721
12.298.2.53 sub()	721
12.298.2.54 subin()	721
12.298.2.55 mul()	721
12.298.2.56 fmadd()	721
12.298.2.57 fmaddin()	721
12.298.2.58 fnmadd()	721
12.298.2.59 fnmaddin()	722
12.298.2.60 fmsub()	722

12.298.2.61 fmsubin()	722
12.298.2.62 eq()	722
12.298.2.63 round()	722
12.298.2.64 mask_high()	722
12.298.2.65 mulhi_fast()	722
12.298.2.66 mod()	722
12.298.2.67 signbits()	723
12.298.2.68 zero()	723
12.298.2.69 vor()	723
12.298.2.70 vxor()	723
12.298.2.71 vand()	723
12.298.2.72 vandnot()	723
12.298.3 Field Documentation	723
12.298.3.1 vect_size	723
12.298.3.2 alignment	723
12.299 Simd512_impl< true, true, true, 8 > Struct Reference	723
12.299.1 Member Typedef Documentation	725
12.299.1.1 vect_t	725
12.299.1.2 half_t	725
12.299.1.3 scalar_t	725
12.299.1.4 simdHalf	726
12.299.1.5 aligned_allocator	726
12.299.1.6 aligned_vector	726
12.299.1.7 is_same_element	726
12.299.2 Member Function Documentation	726
12.299.2.1 type_string()	726
12.299.2.2 valid()	726
12.299.2.3 compliant()	726
12.299.2.4 set1()	726
12.299.2.5 set() [1/2]	726
12.299.2.6 set() [2/2]	726
12.299.2.7 gather()	727
12.299.2.8 load()	727
12.299.2.9 loadu()	727
12.299.2.10 store()	727
12.299.2.11 maskstore()	727
12.299.2.12 storeu()	727
12.299.2.13 stream()	727
12.299.2.14 sll()	727
12.299.2.15 srl()	727
12.299.2.16 sra()	727
12.299.2.17 shuffle()	728

12.299.2.18 unpacklo_intrinsic()	728
12.299.2.19 unpackhi_intrinsic()	728
12.299.2.20 unpacklo()	728
12.299.2.21 unpackhi()	728
12.299.2.22 unpacklohi()	728
12.299.2.23 pack_even()	728
12.299.2.24 pack_odd()	728
12.299.2.25 pack()	728
12.299.2.26 transpose()	729
12.299.2.27 blend()	729
12.299.2.28 add()	729
12.299.2.29 addin()	729
12.299.2.30 sub()	729
12.299.2.31 subin()	729
12.299.2.32 mullo()	729
12.299.2.33 mul()	729
12.299.2.34 mulhi()	729
12.299.2.35 mulx()	730
12.299.2.36 fmadd()	730
12.299.2.37 fmaddin()	730
12.299.2.38 fmaddx()	730
12.299.2.39 fmaddxin()	730
12.299.2.40 fnmadd()	730
12.299.2.41 fnmaddin()	730
12.299.2.42 fnmaddx()	730
12.299.2.43 fnmaddxin()	730
12.299.2.44 fmsub()	731
12.299.2.45 fmsubin()	731
12.299.2.46 fmsubx()	731
12.299.2.47 fmsubxin()	731
12.299.2.48 eq()	731
12.299.2.49 greater()	731
12.299.2.50 lesser()	731
12.299.2.51 greater_eq()	731
12.299.2.52 lesser_eq()	731
12.299.2.53 hadd_to_scal()	732
12.299.2.54 round()	732
12.299.2.55 mask_high()	732
12.299.2.56 mulhi_fast()	732
12.299.2.57 mod()	732
12.299.2.58 signbits()	732
12.299.2.59 zero()	732

12.299.2.60 vor()	732
12.299.2.61 vxor()	732
12.299.2.62 vand()	732
12.299.2.63 vandnot()	733
12.299.3 Field Documentation	733
12.299.3.1 vect_size	733
12.299.3.2 alignment	733
12.300 Simd512i_base Struct Reference	733
12.300.1 Member Typedef Documentation	733
12.300.1.1 vect_t	733
12.300.2 Member Function Documentation	733
12.300.2.1 zero()	733
12.300.2.2 vor()	733
12.300.2.3 vxor()	734
12.300.2.4 vand()	734
12.300.2.5 vandnot()	734
12.301 SimdChooser< T, bool, bool > Struct Template Reference	734
12.302 SimdChooser< T, false, b > Struct Template Reference	734
12.302.1 Member Typedef Documentation	734
12.302.1.1 value	734
12.303 SimdChooser< T, true, false > Struct Template Reference	734
12.303.1 Member Typedef Documentation	735
12.303.1.1 value	735
12.304 SimdChooser< T, true, true > Struct Template Reference	735
12.304.1 Member Typedef Documentation	735
12.304.1.1 value	735
12.305 simdToType< T > Struct Template Reference	735
12.306 Single Struct Reference	735
12.307 Sparse< Field, SparseMatrix_t, IdxT, PtrT > Struct Template Reference	735
12.308 Sparse< _Field, SparseMatrix_t::COO > Struct Template Reference	735
12.308.1 Member Typedef Documentation	736
12.308.1.1 Field	736
12.308.2 Field Documentation	736
12.308.2.1 col	736
12.308.2.2 row	736
12.308.2.3 dat	736
12.308.2.4 delayed	736
12.308.2.5 kmax	736
12.308.2.6 m	736
12.308.2.7 n	736
12.308.2.8 nnz	737
12.308.2.9 nElements	737

12.308.2.10 maxrow	737
12.309 Sparse< _Field, SparseMatrix_t::COO_ZO > Struct Template Reference	737
12.309.1 Member Typedef Documentation	737
12.309.1.1 Field	737
12.309.2 Field Documentation	738
12.309.2.1 cst	738
12.309.2.2 col	738
12.309.2.3 row	738
12.309.2.4 dat	738
12.309.2.5 delayed	738
12.309.2.6 kmax	738
12.309.2.7 m	738
12.309.2.8 n	738
12.309.2.9 nnz	738
12.309.2.10 nElements	738
12.309.2.11 maxrow	738
12.310 Sparse< _Field, SparseMatrix_t::CSR > Struct Template Reference	739
12.310.1 Member Typedef Documentation	739
12.310.1.1 Field	739
12.310.2 Field Documentation	739
12.310.2.1 delayed	739
12.310.2.2 kmax	739
12.310.2.3 m	739
12.310.2.4 n	739
12.310.2.5 nnz	740
12.310.2.6 nElements	740
12.310.2.7 maxrow	740
12.310.2.8 col	740
12.310.2.9 st	740
12.310.2.10 stend	740
12.310.2.11 dat	740
12.311 Sparse< _Field, SparseMatrix_t::CSR_HYB > Struct Template Reference	740
12.311.1 Member Typedef Documentation	741
12.311.1.1 Field	741
12.311.2 Field Documentation	741
12.311.2.1 delayed	741
12.311.2.2 col	741
12.311.2.3 st	741
12.311.2.4 dat	741
12.311.2.5 kmax	741
12.311.2.6 m	741
12.311.2.7 n	741

12.311.2.8 nnz	741
12.311.2.9 nElements	741
12.311.2.10 maxrow	742
12.311.2.11 nOnes	742
12.311.2.12 nMOnes	742
12.311.2.13 nOthers	742
12.312 Sparse< _Field, SparseMatrix_t::CSR_ZO > Struct Template Reference	742
12.312.1 Member Typedef Documentation	743
12.312.1.1 Field	743
12.312.2 Field Documentation	743
12.312.2.1 cst	743
12.312.2.2 delayed	743
12.312.2.3 kmax	743
12.312.2.4 m	743
12.312.2.5 n	743
12.312.2.6 nnz	743
12.312.2.7 nElements	743
12.312.2.8 maxrow	743
12.312.2.9 col	743
12.312.2.10 st	743
12.312.2.11 stend	744
12.312.2.12 dat	744
12.313 Sparse< _Field, SparseMatrix_t::ELL > Struct Template Reference	744
12.313.1 Member Typedef Documentation	744
12.313.1.1 Field	744
12.313.2 Field Documentation	744
12.313.2.1 delayed	744
12.313.2.2 kmax	745
12.313.2.3 m	745
12.313.2.4 n	745
12.313.2.5 ld	745
12.313.2.6 nnz	745
12.313.2.7 nElements	745
12.313.2.8 maxrow	745
12.313.2.9 col	745
12.313.2.10 dat	745
12.314 Sparse< _Field, SparseMatrix_t::ELL_simd > Struct Template Reference	745
12.314.1 Field Documentation	746
12.314.1.1 delayed	746
12.314.1.2 chunk	746
12.314.1.3 m	746
12.314.1.4 n	746



12.314.1.5 ld	746
12.314.1.6 kmax	746
12.314.1.7 nnz	746
12.314.1.8 nElements	746
12.314.1.9 maxrow	747
12.314.1.10 nChunks	747
12.314.1.11 col	747
12.314.1.12 dat	747
12.315 Sparse< _Field, SparseMatrix_t::ELL_simd_ZO > Struct Template Reference	747
12.315.1 Field Documentation	747
12.315.1.1 cst	747
12.315.1.2 delayed	748
12.315.1.3 chunk	748
12.315.1.4 m	748
12.315.1.5 n	748
12.315.1.6 ld	748
12.315.1.7 kmax	748
12.315.1.8 nnz	748
12.315.1.9 nElements	748
12.315.1.10 maxrow	748
12.315.1.11 nChunks	748
12.315.1.12 col	748
12.315.1.13 dat	748
12.316 Sparse< _Field, SparseMatrix_t::ELL_ZO > Struct Template Reference	749
12.316.1 Member Typedef Documentation	749
12.316.1.1 Field	749
12.316.2 Field Documentation	749
12.316.2.1 cst	749
12.316.2.2 delayed	749
12.316.2.3 kmax	749
12.316.2.4 m	749
12.316.2.5 n	750
12.316.2.6 ld	750
12.316.2.7 nnz	750
12.316.2.8 nElements	750
12.316.2.9 maxrow	750
12.316.2.10 col	750
12.316.2.11 dat	750
12.317 Sparse< _Field, SparseMatrix_t::HYB_ZO > Struct Template Reference	750
12.317.1 Member Typedef Documentation	751
12.317.1.1 Field	751
12.317.1.2 Self_t	751

12.317.2 Field Documentation	751
12.317.2.1 delayed	751
12.317.2.2 kmax	751
12.317.2.3 m	751
12.317.2.4 n	751
12.317.2.5 nnz	751
12.317.2.6 maxrow	751
12.317.2.7 nElements	751
12.317.2.8 dat	751
12.317.2.9 one	751
12.317.2.10 mone	752
12.318 Sparse< _Field, SparseMatrix_t::SELL > Struct Template Reference	752
12.318.1 Member Typedef Documentation	752
12.318.1.1 Field	752
12.318.2 Field Documentation	752
12.318.2.1 delayed	752
12.318.2.2 chunk	753
12.318.2.3 kmax	753
12.318.2.4 m	753
12.318.2.5 n	753
12.318.2.6 maxrow	753
12.318.2.7 sigma	753
12.318.2.8 nChunks	753
12.318.2.9 nnz	753
12.318.2.10 nElements	753
12.318.2.11 perm	753
12.318.2.12 st	753
12.318.2.13 chunkSize	753
12.318.2.14 col	754
12.318.2.15 dat	754
12.319 Sparse< _Field, SparseMatrix_t::SELL_ZO > Struct Template Reference	754
12.319.1 Member Typedef Documentation	754
12.319.1.1 Field	754
12.319.2 Field Documentation	755
12.319.2.1 cst	755
12.319.2.2 delayed	755
12.319.2.3 chunk	755
12.319.2.4 kmax	755
12.319.2.5 m	755
12.319.2.6 n	755
12.319.2.7 maxrow	755
12.319.2.8 sigma	755

12.319.2.9 nChunks	755
12.319.2.10 nnz	755
12.319.2.11 nElements	755
12.319.2.12 perm	755
12.319.2.13 st	756
12.319.2.14 chunkSize	756
12.319.2.15 col	756
12.319.2.16 dat	756
12.320 SpMat< Field, flag > Struct Template Reference	756
12.320.1 Field Documentation	756
12.320.1.1 _coo	756
12.320.1.2 _csr	756
12.320.1.3 _ell	756
12.321 StatsMatrix Struct Reference	756
12.321.1 Field Documentation	757
12.321.1.1 rowdim	757
12.321.1.2 coldim	757
12.321.1.3 nOnes	757
12.321.1.4 nMOnes	757
12.321.1.5 nOthers	757
12.321.1.6 nnz	757
12.321.1.7 maxRow	758
12.321.1.8 minRow	758
12.321.1.9 averageRow	758
12.321.1.10 deviationRow	758
12.321.1.11 maxCol	758
12.321.1.12 minCol	758
12.321.1.13 averageCol	758
12.321.1.14 deviationCol	758
12.321.1.15 minColDifference	758
12.321.1.16 maxColDifference	758
12.321.1.17 averageColDifference	758
12.321.1.18 deviationColDifference	758
12.321.1.19 minRowDifference	758
12.321.1.20 maxRowDifference	758
12.321.1.21 averageRowDifference	758
12.321.1.22 deviationRowDifference	759
12.321.1.23 nDenseRows	759
12.321.1.24 nDenseCols	759
12.321.1.25 nEmptyRows	759
12.321.1.26 nEmptyCols	759
12.321.1.27 nEmptyColsEnd	759

12.321.1.28 denseRows	759
12.321.1.29 denseCols	759
12.322 support_fast_mod< T > Struct Template Reference	759
12.323 support_fast_mod< double > Struct Reference	759
12.324 support_fast_mod< float > Struct Reference	760
12.325 support_fast_mod< int64_t > Struct Reference	760
12.326 support_simd< T > Struct Template Reference	760
12.327 support_simd_add< T > Struct Template Reference	761
12.328 support_simd_mod< T > Struct Template Reference	761
12.329 Test< Elt > Class Template Reference	761
12.329.1 Member Typedef Documentation	762
12.329.1.1 Field	762
12.329.1.2 Elt_ptr	762
12.329.1.3 Residu	762
12.329.1.4 enable_if_t	762
12.329.1.5 is_same_element	762
12.329.1.6 enable_if_no_simd_t	762
12.329.1.7 enable_if_simd128_t	762
12.329.1.8 enable_if_simd256_t	763
12.329.1.9 enable_if_simd512_t	763
12.329.2 Constructor & Destructor Documentation	763
12.329.2.1 Test()	763
12.329.3 Member Function Documentation	763
12.329.3.1 cardinality() [1/2]	763
12.329.3.2 cardinality() [2/2]	763
12.329.3.3 test_ftranspose()	763
12.329.3.4 doTests()	763
12.329.3.5 run()	763
12.329.4 Field Documentation	764
12.329.4.1 F	764
12.329.4.2 _mm	764
12.329.4.3 _nn	764
12.330 TestOneMethod< Simd > Class Template Reference	764
12.330.1 Member Typedef Documentation	765
12.330.1.1 Element	765
12.330.1.2 vect_t	765
12.330.1.3 vectElt	765
12.330.1.4 enable_if_t	765
12.330.2 Constructor & Destructor Documentation	765
12.330.2.1 TestOneMethod()	765
12.330.3 Member Function Documentation	765
12.330.3.1 evaluate_scalar_method() [1/3]	765

12.330.3.2 evaluate_scalar_method() [2/3]	766
12.330.3.3 evaluate_scalar_method() [3/3]	766
12.330.3.4 evaluate_simd_method() [1/2]	766
12.330.3.5 evaluate_simd_method() [2/2]	766
12.330.3.6 getStatus()	766
12.330.3.7 getTestName()	766
12.330.3.8 writeResultLine()	766
12.330.3.9 writeDebugData()	766
12.330.4 Field Documentation	766
12.330.4.1 vect_size	766
12.330.4.2 nb_lref	767
12.330.4.3 name	767
12.330.4.4 inputs	767
12.330.4.5 outputs_simd	767
12.330.4.6 outputs_scalar	767
12.331 tfn_minus Struct Reference	767
12.331.1 Member Function Documentation	767
12.331.1.1 operator>()	767
12.332 tfn_minus_eq Struct Reference	767
12.332.1 Member Function Documentation	768
12.332.1.1 operator>()	768
12.333 tfn_mul Struct Reference	768
12.333.1 Member Function Documentation	768
12.333.1.1 operator>()	768
12.334 tfn_mul_eq Struct Reference	768
12.334.1 Member Function Documentation	768
12.334.1.1 operator>()	768
12.335 tfn_plus Struct Reference	768
12.335.1 Member Function Documentation	769
12.335.1.1 operator>()	769
12.336 tfn_plus_eq Struct Reference	769
12.336.1 Member Function Documentation	769
12.336.1.1 operator>()	769
12.337 Threads Struct Reference	769
12.338 ThreeD Struct Reference	769
12.339 ThreeDAdaptive Struct Reference	769
12.340 ThreeDInPlace Struct Reference	770
12.341 TRSMHelper< ReclterTrait, ParSeqTrait > Struct Template Reference	770
12.341.1 Detailed Description	770
12.341.2 Constructor & Destructor Documentation	770
12.341.2.1 TRSMHelper() [1/3]	770
12.341.2.2 TRSMHelper() [2/3]	770

12.341.2.3 TRSMHelper() [3/3]	770
12.341.3 Member Function Documentation	771
12.341.3.1 pMMH() [1/2]	771
12.341.3.2 pMMH() [2/2]	771
12.341.4 Field Documentation	771
12.341.4.1 parseq	771
12.342 TwoD Struct Reference	771
12.343 TwoDAdaptive Struct Reference	771
12.344 UnparametricTag Struct Reference	771
12.344.1 Detailed Description	771
12.345 width< T > Struct Template Reference	772
12.345.1 Field Documentation	772
12.345.1.1 value	772
12.346 width< double > Struct Reference	772
12.346.1 Field Documentation	772
12.346.1.1 value	772
12.347 width< float > Struct Reference	772
12.347.1 Field Documentation	772
12.347.1.1 value	772
12.348 Winograd Struct Reference	772
12.349 WinogradPar Struct Reference	772
<b>13 File Documentation</b>	<b>773</b>
13.1 arithprog.C File Reference	773
13.1.1 Typedef Documentation	773
13.1.1.1 TTimer	773
13.1.2 Function Documentation	773
13.1.2.1 main()	773
13.2 autotune/charpoly.C File Reference	773
13.2.1 Macro Definition Documentation	774
13.2.1.1 CUBE	774
13.2.1.2 GFOPS	774
13.2.2 Typedef Documentation	774
13.2.2.1 TTimer	774
13.2.3 Function Documentation	774
13.2.3.1 main()	774
13.3 examples/charpoly.C File Reference	774
13.3.1 Function Documentation	774
13.3.1.1 main()	774
13.4 fsyrk.C File Reference	775
13.4.1 Macro Definition Documentation	775
13.4.1.1 CUBE	775

13.4.1.2 GFOPS	775
13.4.2 Function Documentation	775
13.4.2.1 main()	775
13.5 fsytrf.C File Reference	775
13.5.1 Macro Definition Documentation	776
13.5.1.1 CUBE	776
13.5.1.2 GFOPS	776
13.5.2 Function Documentation	776
13.5.2.1 main()	776
13.6 ftrtri.C File Reference	776
13.6.1 Macro Definition Documentation	776
13.6.1.1 CUBE	776
13.6.1.2 GFOPS	777
13.6.2 Function Documentation	777
13.6.2.1 main()	777
13.7 autotune/pluq.C File Reference	777
13.7.1 Macro Definition Documentation	777
13.7.1.1 CUBE	777
13.7.1.2 GFOPS	777
13.7.2 Function Documentation	777
13.7.2.1 main()	777
13.8 examples/pluq.C File Reference	778
13.8.1 Function Documentation	778
13.8.1.1 main()	778
13.9 winograd.C File Reference	778
13.9.1 Macro Definition Documentation	778
13.9.1.1 DOUBLE_TO_FLOAT_CROSSOVER	778
13.9.1.2 GFOPS	778
13.9.2 Function Documentation	779
13.9.2.1 balanced() [1/2]	779
13.9.2.2 balanced() [2/2]	779
13.9.2.3 main()	779
13.10 benchmark-charpoly-mp.C File Reference	779
13.10.1 Macro Definition Documentation	779
13.10.1.1 __FFLASFFPACK_FORCE_SEQ	779
13.10.2 Function Documentation	779
13.10.2.1 main()	779
13.11 benchmark-charpoly.C File Reference	779
13.11.1 Macro Definition Documentation	780
13.11.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	780
13.11.2 Function Documentation	780
13.11.2.1 run_with_field()	780

13.11.2.2 main()	780
13.12 benchmark-checkers.C File Reference	780
13.12.1 Macro Definition Documentation	781
13.12.1.1 ENABLE_ALL_CHECKINGS	781
13.12.1.2 _NR_TESTS	781
13.12.1.3 _MAX_SIZE_MATRICES	781
13.12.1.4 CUBE	781
13.12.2 Function Documentation	781
13.12.2.1 main()	781
13.13 benchmark-dgemm.C File Reference	781
13.13.1 Macro Definition Documentation	782
13.13.1.1 CBLAS_GEMM	782
13.13.2 Typedef Documentation	782
13.13.2.1 TTimer	782
13.13.2.2 Floats	782
13.13.3 Function Documentation	782
13.13.3.1 main()	782
13.14 benchmark-dgetrf.C File Reference	782
13.14.1 Macro Definition Documentation	782
13.14.1.1 __FFLASFFPACK_HAVE_DGETRF	782
13.14.2 Function Documentation	783
13.14.2.1 main()	783
13.15 benchmark-dgetri.C File Reference	783
13.15.1 Function Documentation	783
13.15.1.1 main()	783
13.16 benchmark-dsytrf.C File Reference	783
13.16.1 Macro Definition Documentation	783
13.16.1.1 EFFGFF	783
13.16.2 Function Documentation	784
13.16.2.1 main()	784
13.17 benchmark-dtrsm.C File Reference	784
13.17.1 Function Documentation	784
13.17.1.1 main()	784
13.18 benchmark-dtrtri.C File Reference	784
13.18.1 Macro Definition Documentation	785
13.18.1.1 __FFLASFFPACK_HAVE_DTRTRI	785
13.18.2 Function Documentation	785
13.18.2.1 main()	785
13.19 benchmark-fadd-lvl2.C File Reference	785
13.19.1 Macro Definition Documentation	785
13.19.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	785
13.19.2 Function Documentation	785



13.19.2.1 main()	785
13.20 benchmark-fdot.C File Reference	785
13.20.1 Macro Definition Documentation	786
13.20.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	786
13.20.2 Function Documentation	786
13.20.2.1 run_with_field()	786
13.20.2.2 main()	786
13.21 benchmark-fgemm-mp.C File Reference	786
13.21.1 Macro Definition Documentation	787
13.21.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	787
13.21.1.2 MG_DEFAULT	787
13.21.1.3 STD_RECINT_SIZE	787
13.21.2 Function Documentation	787
13.21.2.1 tmain()	787
13.21.2.2 main()	787
13.22 benchmark-fgemm-rns.C File Reference	787
13.22.1 Macro Definition Documentation	788
13.22.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	788
13.22.2 Typedef Documentation	788
13.22.2.1 RNS	788
13.22.2.2 Field	788
13.22.2.3 Element_ptr	788
13.22.2.4 ConstElement_ptr	788
13.22.2.5 THREADS	788
13.22.2.6 GRAIN	788
13.22.2.7 TWOD	788
13.22.2.8 TWODA	788
13.22.2.9 THREEED	788
13.22.2.10 THREEEDA	788
13.22.2.11 THREEEDIP	789
13.22.2.12 PSeq	789
13.22.3 Function Documentation	789
13.22.3.1 main()	789
13.23 benchmark-fgemm.C File Reference	789
13.23.1 Macro Definition Documentation	789
13.23.1.1 CLASSIC_HYBRID	789
13.23.2 Function Documentation	789
13.23.2.1 main()	789
13.24 benchmark-fgemv-mp.C File Reference	789
13.24.1 Macro Definition Documentation	790
13.24.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	790
13.24.1.2 MG_DEFAULT	790

13.24.1.3	STD_RECINT_SIZE	790
13.24.2	Function Documentation	790
13.24.2.1	write_matrix()	790
13.24.2.2	tmain()	790
13.24.2.3	main()	790
13.25	benchmark-fgemv.C File Reference	791
13.25.1	Macro Definition Documentation	791
13.25.1.1	__FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	791
13.25.2	Function Documentation	792
13.25.2.1	fill_value()	792
13.25.2.2	genData()	792
13.25.2.3	check_result()	792
13.25.2.4	benchmark_with_timer()	792
13.25.2.5	benchmark_disp()	793
13.25.2.6	benchmark_in_Field()	793
13.25.2.7	benchmark_with_field() [1/2]	793
13.25.2.8	benchmark_with_field() [2/2]	793
13.25.2.9	main()	794
13.26	benchmark-fgesv.C File Reference	794
13.26.1	Macro Definition Documentation	794
13.26.1.1	__FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	794
13.26.2	Function Documentation	794
13.26.2.1	main()	794
13.27	benchmark-fsyr2k.C File Reference	794
13.27.1	Function Documentation	795
13.27.1.1	main()	795
13.28	benchmark-fsyrk.C File Reference	795
13.28.1	Macro Definition Documentation	795
13.28.1.1	__FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	795
13.28.2	Function Documentation	795
13.28.2.1	main()	795
13.29	benchmark-fsytrf.C File Reference	795
13.29.1	Macro Definition Documentation	796
13.29.1.1	__FFPACK_FSYTRF_BC_CROUT	796
13.29.1.2	__FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	796
13.29.1.3	CUBE	796
13.29.2	Function Documentation	796
13.29.2.1	main()	796
13.30	benchmark-ftsm-mp.C File Reference	796
13.30.1	Macro Definition Documentation	796
13.30.1.1	__FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	796
13.30.2	Function Documentation	797

13.30.2.1 main()	797
13.31 benchmark-ftsrm.C File Reference	797
13.31.1 Macro Definition Documentation	797
13.31.1.1 __FflasFfpack_OPENBLAS_NT_ALREADY_SET	797
13.31.2 Function Documentation	797
13.31.2.1 main()	797
13.32 benchmark-ftsrv.C File Reference	797
13.32.1 Macro Definition Documentation	798
13.32.1.1 __FflasFfpack_OPENBLAS_NT_ALREADY_SET	798
13.32.2 Function Documentation	798
13.32.2.1 main()	798
13.33 benchmark-ftsrti.C File Reference	798
13.33.1 Macro Definition Documentation	798
13.33.1.1 __FflasFfpack_OPENBLAS_NT_ALREADY_SET	798
13.33.1.2 CUBE	798
13.33.2 Function Documentation	798
13.33.2.1 main()	798
13.34 benchmark-inverse.C File Reference	798
13.34.1 Macro Definition Documentation	799
13.34.1.1 CUBE	799
13.34.2 Function Documentation	799
13.34.2.1 main()	799
13.35 benchmark-lqup-mp.C File Reference	799
13.35.1 Function Documentation	799
13.35.1.1 main()	799
13.36 benchmark-lqup.C File Reference	800
13.36.1 Macro Definition Documentation	800
13.36.1.1 CUBE	800
13.36.2 Function Documentation	800
13.36.2.1 main()	800
13.37 benchmark-pluq.C File Reference	800
13.37.1 Macro Definition Documentation	801
13.37.1.1 __FflasFfpack_OPENBLAS_NT_ALREADY_SET	801
13.37.1.2 CUBE	801
13.37.2 Typedef Documentation	801
13.37.2.1 Field	801
13.37.3 Function Documentation	801
13.37.3.1 verification_PLUQ()	801
13.37.3.2 Rec_Initialize()	801
13.37.3.3 main()	801
13.38 benchmark-quasisep.C File Reference	801
13.38.1 Macro Definition Documentation	802

13.38.1.1 <a href="#">__FFLASFFPACK_OPENBLAS_NT_ALREADY_SET</a>	802
13.38.2 Function Documentation	802
13.38.2.1 <a href="#">run_with_field()</a>	802
13.38.2.2 <a href="#">main()</a>	802
13.39 benchmark-storage-transpose.C File Reference	802
13.39.1 Function Documentation	803
13.39.1.1 <a href="#">main()</a>	803
13.40 benchmark-wino.C File Reference	803
13.40.1 Macro Definition Documentation	803
13.40.1.1 <a href="#">CUBE</a>	803
13.40.2 Function Documentation	803
13.40.2.1 <a href="#">launch_wino()</a>	803
13.40.2.2 <a href="#">main()</a>	804
13.41 mainpage.doxy File Reference	804
13.42 det.C File Reference	804
13.42.1 Function Documentation	804
13.42.1.1 <a href="#">main()</a>	804
13.43 matmul.C File Reference	804
13.43.1 Function Documentation	804
13.43.1.1 <a href="#">main()</a>	804
13.44 rank.C File Reference	805
13.44.1 Function Documentation	805
13.44.1.1 <a href="#">main()</a>	805
13.45 solve.C File Reference	805
13.45.1 Function Documentation	805
13.45.1.1 <a href="#">main()</a>	805
13.46 checker_charpoly.inl File Reference	805
13.46.1 Macro Definition Documentation	806
13.46.1.1 <a href="#">__FFLASFFPACK_checker_charpoly_INL</a>	806
13.47 checker_det.inl File Reference	806
13.47.1 Macro Definition Documentation	806
13.47.1.1 <a href="#">__FFLASFFPACK_checker_det_INL</a>	806
13.48 checker_empty.h File Reference	806
13.49 checker_fgemm.inl File Reference	806
13.49.1 Macro Definition Documentation	807
13.49.1.1 <a href="#">__FFLASFFPACK_checker_fgemm_INL</a>	807
13.50 checker_ftsm.inl File Reference	807
13.50.1 Macro Definition Documentation	807
13.50.1.1 <a href="#">__FFLASFFPACK_checker_ftsm_INL</a>	807
13.51 checker_invert.inl File Reference	807
13.51.1 Macro Definition Documentation	807
13.51.1.1 <a href="#">__FFLASFFPACK_checker_invert_INL</a>	807

13.52 checker_pluq.inl File Reference . . . . .	807
13.52.1 Macro Definition Documentation . . . . .	808
13.52.1.1 __FFLASFFPACK_checker_pluq_INL . . . . .	808
13.53 checkers.doxy File Reference . . . . .	808
13.54 checkers_fflas.h File Reference . . . . .	808
13.55 checkers_fflas.inl File Reference . . . . .	808
13.55.1 Macro Definition Documentation . . . . .	809
13.55.1.1 FFLASFFPACK_checkers_fflas_inl_H . . . . .	809
13.56 checkers_ffpack.h File Reference . . . . .	809
13.57 checkers_ffpack.inl File Reference . . . . .	809
13.57.1 Macro Definition Documentation . . . . .	810
13.57.1.1 FFLASFFPACK_checkers_ffpack_inl_H . . . . .	810
13.58 config-blas.h File Reference . . . . .	810
13.58.1 Macro Definition Documentation . . . . .	811
13.58.1.1 CBLAS_INT . . . . .	811
13.58.1.2 CBLAS_ENUM_DEFINED_H . . . . .	811
13.58.1.3 CBLAS_EXTERNALS . . . . .	811
13.58.1.4 blas_enum . . . . .	811
13.58.2 Enumeration Type Documentation . . . . .	811
13.58.2.1 CBLAS_ORDER . . . . .	811
13.58.2.2 CBLAS_TRANSPOSE . . . . .	811
13.58.2.3 CBLAS_UPLO . . . . .	812
13.58.2.4 CBLAS_DIAG . . . . .	812
13.58.2.5 CBLAS_SIDE . . . . .	812
13.58.3 Function Documentation . . . . .	812
13.58.3.1 daxpy_() . . . . .	812
13.58.3.2 saxpy_() . . . . .	812
13.58.3.3 ddot_() . . . . .	813
13.58.3.4 sdot_() . . . . .	813
13.58.3.5 dasum_() . . . . .	813
13.58.3.6 idamax_() . . . . .	813
13.58.3.7 dnrm2_() . . . . .	813
13.58.3.8 dgemv_() . . . . .	813
13.58.3.9 sgemv_() . . . . .	813
13.58.3.10 dger_() . . . . .	814
13.58.3.11 sger_() . . . . .	814
13.58.3.12 dcopy_() . . . . .	814
13.58.3.13 scopy_() . . . . .	814
13.58.3.14 dscal_() . . . . .	814
13.58.3.15 sscal_() . . . . .	815
13.58.3.16 dtrsm_() . . . . .	815
13.58.3.17 strsm_() . . . . .	815

13.58.3.18 dtrmm_()	815
13.58.3.19 strmm_()	815
13.58.3.20 sgemm_()	816
13.58.3.21 dgemm_()	816
13.58.3.22 cblas_dsyrk()	816
13.59 config.h File Reference	817
13.59.1 Macro Definition Documentation	817
13.59.1.1 HAVE_BLAS	817
13.59.1.2 HAVE_CBLAS	817
13.59.1.3 HAVE_CXX11	818
13.59.1.4 HAVE_DLFCN_H	818
13.59.1.5 HAVE_FLOAT_H	818
13.59.1.6 HAVE_INT128	818
13.59.1.7 HAVE_INTPTR_T	818
13.59.1.8 HAVE_LAPACK	818
13.59.1.9 HAVE_LIMITS_H	818
13.59.1.10 HAVE_LITTLE_ENDIAN	818
13.59.1.11 HAVE_PTHREAD_H	818
13.59.1.12 HAVE_STDDEF_H	818
13.59.1.13 HAVE_STDINT_H	818
13.59.1.14 HAVE_STDIO_H	818
13.59.1.15 HAVE_STDLIB_H	818
13.59.1.16 HAVE_STRINGS_H	818
13.59.1.17 HAVE_STRING_H	818
13.59.1.18 HAVE_SYS_STAT_H	819
13.59.1.19 HAVE_SYS_TIME_H	819
13.59.1.20 HAVE_SYS_TYPES_H	819
13.59.1.21 HAVE_UNISTD_H	819
13.59.1.22 LT_OBJDIR	819
13.59.1.23 OPENBLAS_NUM_THREADS	819
13.59.1.24 PACKAGE	819
13.59.1.25 PACKAGE_BUGREPORT	819
13.59.1.26 PACKAGE_NAME	819
13.59.1.27 PACKAGE_STRING	819
13.59.1.28 PACKAGE_TARNAME	819
13.59.1.29 PACKAGE_URL	819
13.59.1.30 PACKAGE_VERSION	819
13.59.1.31 SIZEOF_CHAR	819
13.59.1.32 SIZEOF_INT	819
13.59.1.33 SIZEOF_LONG	820
13.59.1.34 SIZEOF_LONG_LONG	820
13.59.1.35 SIZEOF_SHORT	820

13.59.1.36	SIZEOF___INT64_T	820
13.59.1.37	STDC_HEADERS	820
13.59.1.38	USE_OPENMP	820
13.59.1.39	VERSION	820
13.60	fflas-ffpack/config.h File Reference	820
13.60.1	Macro Definition Documentation	821
13.60.1.1	__FFLASFFPACK_HAVE_BLAS	821
13.60.1.2	__FFLASFFPACK_HAVE_CBLAS	821
13.60.1.3	__FFLASFFPACK_HAVE_CXX11	821
13.60.1.4	__FFLASFFPACK_HAVE_DLFCN_H	821
13.60.1.5	__FFLASFFPACK_HAVE_FLOAT_H	821
13.60.1.6	__FFLASFFPACK_HAVE_INT128	821
13.60.1.7	__FFLASFFPACK_HAVE_INTPYPES_H	821
13.60.1.8	__FFLASFFPACK_HAVE_LAPACK	821
13.60.1.9	__FFLASFFPACK_HAVE_LIMITS_H	821
13.60.1.10	__FFLASFFPACK_HAVE_LITTLE_ENDIAN	821
13.60.1.11	__FFLASFFPACK_HAVE_PTHREAD_H	821
13.60.1.12	__FFLASFFPACK_HAVE_STDDEF_H	822
13.60.1.13	__FFLASFFPACK_HAVE_STDINT_H	822
13.60.1.14	__FFLASFFPACK_HAVE_STDIO_H	822
13.60.1.15	__FFLASFFPACK_HAVE_STDLIB_H	822
13.60.1.16	__FFLASFFPACK_HAVE_STRINGS_H	822
13.60.1.17	__FFLASFFPACK_HAVE_STRING_H	822
13.60.1.18	__FFLASFFPACK_HAVE_SYS_STAT_H	822
13.60.1.19	__FFLASFFPACK_HAVE_SYS_TIME_H	822
13.60.1.20	__FFLASFFPACK_HAVE_SYS_TYPES_H	822
13.60.1.21	__FFLASFFPACK_HAVE_UNISTD_H	822
13.60.1.22	__FFLASFFPACK_LT_OBJDIR	822
13.60.1.23	__FFLASFFPACK_OPENBLAS_NUM_THREADS	822
13.60.1.24	__FFLASFFPACK_PACKAGE	822
13.60.1.25	__FFLASFFPACK_PACKAGE_BUGREPORT	822
13.60.1.26	__FFLASFFPACK_PACKAGE_NAME	822
13.60.1.27	__FFLASFFPACK_PACKAGE_STRING	823
13.60.1.28	__FFLASFFPACK_PACKAGE_TARNAME	823
13.60.1.29	__FFLASFFPACK_PACKAGE_URL	823
13.60.1.30	__FFLASFFPACK_PACKAGE_VERSION	823
13.60.1.31	__FFLASFFPACK_SIZEOF_CHAR	823
13.60.1.32	__FFLASFFPACK_SIZEOF_INT	823
13.60.1.33	__FFLASFFPACK_SIZEOF_LONG	823
13.60.1.34	__FFLASFFPACK_SIZEOF_LONG_LONG	823
13.60.1.35	__FFLASFFPACK_SIZEOF_SHORT	823
13.60.1.36	__FFLASFFPACK_SIZEOF___INT64_T	823

13.60.1.37	<a href="#">__FFLASFFPACK_STDC_HEADERS</a>	823
13.60.1.38	<a href="#">__FFLASFFPACK_USE_OPENMP</a>	823
13.60.1.39	<a href="#">__FFLASFFPACK_VERSION</a>	823
13.61	<a href="#">fflas-ffpack-config.h File Reference</a>	823
13.61.1	<a href="#">Detailed Description</a>	824
13.61.2	<a href="#">Macro Definition Documentation</a>	824
13.61.2.1	<a href="#">GCC_VERSION</a>	824
13.62	<a href="#">fflas-ffpack-default-thresholds.h File Reference</a>	824
13.62.1	<a href="#">Macro Definition Documentation</a>	824
13.62.1.1	<a href="#">__FFLASFFPACK_WINOTHRESHOLD</a>	824
13.62.1.2	<a href="#">__FFLASFFPACK_WINOTHRESHOLD_FLT</a>	824
13.62.1.3	<a href="#">__FFLASFFPACK_WINOTHRESHOLD_BAL</a>	824
13.62.1.4	<a href="#">__FFLASFFPACK_WINOTHRESHOLD_BAL_FLT</a>	824
13.62.1.5	<a href="#">__FFLASFFPACK_PLUQ_THRESHOLD</a>	824
13.62.1.6	<a href="#">__FFLASFFPACK_CHARPOLY_LUKrylov_ArithProg_THRESHOLD</a>	824
13.62.1.7	<a href="#">__FFLASFFPACK_CHARPOLY_Danilevskii_LUKrylov_THRESHOLD</a>	825
13.62.1.8	<a href="#">__FFLASFFPACK_ARITHPROG_THRESHOLD</a>	825
13.62.1.9	<a href="#">__FFLASFFPACK_FTRTRI_THRESHOLD</a>	825
13.62.1.10	<a href="#">__FFLASFFPACK_FSYTRF_THRESHOLD</a>	825
13.62.1.11	<a href="#">__FFLASFFPACK_FSYRK_THRESHOLD</a>	825
13.63	<a href="#">fflas-ffpack-thresholds.h File Reference</a>	825
13.64	<a href="#">fflas-ffpack.doxy File Reference</a>	825
13.65	<a href="#">fflas-ffpack.h File Reference</a>	825
13.65.1	<a href="#">Detailed Description</a>	825
13.66	<a href="#">fflas.doxy File Reference</a>	825
13.67	<a href="#">fflas.h File Reference</a>	825
13.67.1	<a href="#">Detailed Description</a>	826
13.67.2	<a href="#">Macro Definition Documentation</a>	826
13.67.2.1	<a href="#">WINOTHRESHOLD</a>	826
13.67.2.2	<a href="#">DOUBLE_TO_FLOAT_CROSSOVER</a>	826
13.68	<a href="#">fflas_bounds.inl File Reference</a>	827
13.68.1	<a href="#">Macro Definition Documentation</a>	827
13.68.1.1	<a href="#">__FFLASFFPACK_fflas_bounds_INL</a>	827
13.68.1.2	<a href="#">FFLAS_INT_TYPE</a>	827
13.69	<a href="#">fflas_enum.h File Reference</a>	827
13.70	<a href="#">fflas_fadd.h File Reference</a>	828
13.71	<a href="#">fflas_fadd.inl File Reference</a>	829
13.71.1	<a href="#">Macro Definition Documentation</a>	830
13.71.1.1	<a href="#">__FFLASFFPACK_fadd_INL</a>	830
13.72	<a href="#">fflas_fassign.h File Reference</a>	830
13.73	<a href="#">fflas_fassign.inl File Reference</a>	831
13.73.1	<a href="#">Macro Definition Documentation</a>	831



13.73.1.1	<a href="#">__FFLASFFPACK_fassign_INL</a>	831
13.74	<a href="#">fflas_faxpy.inl</a> File Reference	831
13.74.1	Macro Definition Documentation	832
13.74.1.1	<a href="#">__FFLASFFPACK_faxpy_INL</a>	832
13.75	<a href="#">fflas_fdot.inl</a> File Reference	832
13.75.1	Macro Definition Documentation	833
13.75.1.1	<a href="#">__FFLASFFPACK_fdot_INL</a>	833
13.76	<a href="#">fflas_fgemm.inl</a> File Reference	833
13.76.1	Macro Definition Documentation	835
13.76.1.1	<a href="#">__FFLASFFPACK_fgemm_INL</a>	835
13.77	<a href="#">fgemm_classical.inl</a> File Reference	835
13.78	<a href="#">fgemm_classical_mp.inl</a> File Reference	835
13.78.1	Detailed Description	837
13.78.2	Macro Definition Documentation	837
13.78.2.1	<a href="#">__FFPACK_fgemm_classical_INL</a>	837
13.79	<a href="#">fgemm_winograd.inl</a> File Reference	837
13.79.1	Macro Definition Documentation	839
13.79.1.1	<a href="#">__FFLASFFPACK_fflas_fflas_fgemm_winograd_INL</a>	839
13.79.1.2	NEWWINO	839
13.80	<a href="#">matmul.doxy</a> File Reference	839
13.81	<a href="#">schedule_bini.inl</a> File Reference	839
13.81.1	Detailed Description	839
13.81.2	Macro Definition Documentation	839
13.81.2.1	<a href="#">__FFLASFFPACK_fgemm_bini_INL</a>	839
13.82	<a href="#">schedule_winograd.inl</a> File Reference	839
13.82.1	Macro Definition Documentation	840
13.82.1.1	<a href="#">__FFLASFFPACK_fgemm_winograd_INL</a>	840
13.83	<a href="#">schedule_winograd_acc.inl</a> File Reference	840
13.83.1	Macro Definition Documentation	841
13.83.1.1	<a href="#">__FFLASFFPACK_fgemm_winograd_acc_INL</a>	841
13.84	<a href="#">schedule_winograd_acc_ip.inl</a> File Reference	841
13.84.1	Macro Definition Documentation	841
13.84.1.1	<a href="#">__FFLASFFPACK_fgemm_winograd_acc_ip_INL</a>	841
13.85	<a href="#">schedule_winograd_ip.inl</a> File Reference	841
13.85.1	Macro Definition Documentation	842
13.85.1.1	<a href="#">__FFLASFFPACK_fgemm_winograd_ip_INL</a>	842
13.86	<a href="#">fflas_fgenv.inl</a> File Reference	842
13.86.1	Macro Definition Documentation	843
13.86.1.1	<a href="#">__FFLASFFPACK_fgenv_INL</a>	843
13.87	<a href="#">fflas_fgenv_mp.inl</a> File Reference	844
13.87.1	Macro Definition Documentation	844
13.87.1.1	<a href="#">__FFLASFFPACK_fgenv_mp_INL</a>	844

13.88 fflas_fger.inl File Reference . . . . .	844
13.88.1 Macro Definition Documentation . . . . .	845
13.88.1.1 __FFLASFFPACK_fger_INL . . . . .	845
13.89 fflas_fger_mp.inl File Reference . . . . .	845
13.89.1 Macro Definition Documentation . . . . .	846
13.89.1.1 __FFPACK_fger_mp_INL . . . . .	846
13.90 fflas_freduce.h File Reference . . . . .	846
13.91 fflas_freduce.inl File Reference . . . . .	847
13.91.1 Macro Definition Documentation . . . . .	849
13.91.1.1 __FFLASFFPACK_fflas_freduce_INL . . . . .	849
13.91.1.2 FFLASFFPACK_COPY_REDUCE . . . . .	849
13.92 fflas_freduce_mp.inl File Reference . . . . .	849
13.92.1 Macro Definition Documentation . . . . .	849
13.92.1.1 __FFLASFFPACK_fflas_freduce_mp_INL . . . . .	849
13.93 fflas_freivalds.inl File Reference . . . . .	849
13.93.1 Macro Definition Documentation . . . . .	849
13.93.1.1 __FFLASFFPACK_freivalds_INL . . . . .	849
13.94 fflas_fscal.h File Reference . . . . .	850
13.95 fflas_fscal.inl File Reference . . . . .	850
13.95.1 Macro Definition Documentation . . . . .	851
13.95.1.1 __FFLASFFPACK_fscal_INL . . . . .	851
13.96 fflas_fscal_mp.inl File Reference . . . . .	851
13.96.1 Macro Definition Documentation . . . . .	852
13.96.1.1 __FFLASFFPACK_fscal_mp_INL . . . . .	852
13.97 fflas_fsyr2k.inl File Reference . . . . .	852
13.97.1 Macro Definition Documentation . . . . .	852
13.97.1.1 __FFLASFFPACK_fflas_fsyr2k_INL . . . . .	852
13.98 fflas_fsyrk.inl File Reference . . . . .	853
13.98.1 Macro Definition Documentation . . . . .	854
13.98.1.1 __FFLASFFPACK_fflas_fsyrk_INL . . . . .	854
13.99 fflas_fsyrk_strassen.inl File Reference . . . . .	854
13.99.1 Macro Definition Documentation . . . . .	855
13.99.1.1 __FFLASFFPACK_fflas_fsyrk_strassen_INL . . . . .	855
13.100 fflas_ftrmm.inl File Reference . . . . .	855
13.100.1 Macro Definition Documentation . . . . .	856
13.100.1.1 __FFLASFFPACK_ftrmm_INL . . . . .	856
13.101 fflas_ftrsm.inl File Reference . . . . .	856
13.101.1 Macro Definition Documentation . . . . .	856
13.101.1.1 __FFLASFFPACK_ftrsm_INL . . . . .	856
13.102 fflas_ftrsm_mp.inl File Reference . . . . .	857
13.102.1 Detailed Description . . . . .	857
13.102.2 Macro Definition Documentation . . . . .	857

13.102.2.1	<a href="#">__FFPACK_ftrsm_mp_INL</a>	857
13.103	<a href="#">fflas_ftrsv.inl</a> File Reference	857
13.103.1	Macro Definition Documentation	858
13.103.1.1	<a href="#">__FFLASFFPACK_ftrsv_INL</a>	858
13.104	<a href="#">fflas_helpers.inl</a> File Reference	858
13.104.1	Macro Definition Documentation	859
13.104.1.1	<a href="#">__FFLASFFPACK_fflas_fflas_mmmhelper_INL</a>	859
13.105	<a href="#">igemm.doxy</a> File Reference	859
13.106	<a href="#">igemm.h</a> File Reference	859
13.107	<a href="#">igemm.inl</a> File Reference	859
13.107.1	Macro Definition Documentation	860
13.107.1.1	<a href="#">__FFLASFFPACK_fflas_igemm_igemm_INL</a>	860
13.108	<a href="#">igemm_kernels.h</a> File Reference	860
13.109	<a href="#">igemm_kernels.inl</a> File Reference	861
13.109.1	Macro Definition Documentation	861
13.109.1.1	<a href="#">__FFLASFFPACK_fflas_igemm_igemm_kernels_INL</a>	861
13.110	<a href="#">igemm_tools.h</a> File Reference	861
13.111	<a href="#">igemm_tools.inl</a> File Reference	862
13.111.1	Macro Definition Documentation	862
13.111.1.1	<a href="#">__FFLASFFPACK_fflas_igemm_igemm_tools_INL</a>	862
13.112	<a href="#">fflas_level1.inl</a> File Reference	862
13.112.1	Macro Definition Documentation	864
13.112.1.1	<a href="#">__FFLASFFPACK_fflas_fflas_level1_INL</a>	864
13.113	<a href="#">fflas_level2.inl</a> File Reference	865
13.113.1	Macro Definition Documentation	867
13.113.1.1	<a href="#">__FFLASFFPACK_fflas_fflas_level2_INL</a>	867
13.114	<a href="#">fflas_level3.inl</a> File Reference	867
13.114.1	Macro Definition Documentation	870
13.114.1.1	<a href="#">__FFLASFFPACK_fflas_fflas_level3_INL</a>	870
13.114.1.2	<a href="#">__FFLAS__TRSM_READONLY</a>	870
13.115	<a href="#">fflas_pfgemm.inl</a> File Reference	870
13.115.1	Macro Definition Documentation	870
13.115.1.1	<a href="#">__FFLASFFPACK_fflas_pfgemm_INL</a>	870
13.115.1.2	<a href="#">__FFLASFFPACK_SEQPARTHRESHOLD</a>	870
13.115.1.3	<a href="#">__FFLASFFPACK_DIMKPENALTY</a>	870
13.116	<a href="#">fflas_pftrsm.inl</a> File Reference	871
13.116.1	Macro Definition Documentation	871
13.116.1.1	<a href="#">__FFLASFFPACK_fflas_pftrsm_INL</a>	871
13.116.1.2	<a href="#">PTRSM_HYBRID_THRESHOLD</a>	871
13.117	<a href="#">fflas_simd.h</a> File Reference	871
13.117.1	Macro Definition Documentation	872
13.117.1.1	<a href="#">SIMD_INT</a>	872

13.117.1.2	<a href="#">INLINE</a>	872
13.117.1.3	<a href="#">CONST</a>	872
13.117.1.4	<a href="#">PURE</a>	872
13.117.1.5	<a href="#">NORML_MOD</a>	872
13.117.1.6	<a href="#">FLOAT_MOD</a>	873
13.117.2	<a href="#">Typedef Documentation</a>	873
13.117.2.1	<a href="#">Simd</a>	873
13.118	<a href="#">simd.doxy File Reference</a>	873
13.119	<a href="#">simd128.inl File Reference</a>	873
13.119.1	<a href="#">Macro Definition Documentation</a>	873
13.119.1.1	<a href="#">__FFLASFFPACK_fflas_ffpack_utils_simd128_INL</a>	873
13.119.2	<a href="#">Typedef Documentation</a>	873
13.119.2.1	<a href="#">Simd128</a>	873
13.120	<a href="#">simd128_double.inl File Reference</a>	874
13.120.1	<a href="#">Macro Definition Documentation</a>	874
13.120.1.1	<a href="#">__FFLASFFPACK_fflas_ffpack_utils_simd128_double_INL</a>	874
13.121	<a href="#">simd128_float.inl File Reference</a>	874
13.121.1	<a href="#">Macro Definition Documentation</a>	874
13.121.1.1	<a href="#">__FFLASFFPACK_fflas_ffpack_utils_simd128_float_INL</a>	874
13.122	<a href="#">simd128_int16.inl File Reference</a>	874
13.122.1	<a href="#">Macro Definition Documentation</a>	875
13.122.1.1	<a href="#">__FFLASFFPACK_fflas_ffpack_utils_simd128_int16_INL</a>	875
13.123	<a href="#">simd128_int32.inl File Reference</a>	875
13.123.1	<a href="#">Macro Definition Documentation</a>	875
13.123.1.1	<a href="#">__FFLASFFPACK_fflas_ffpack_utils_simd128_int32_INL</a>	875
13.124	<a href="#">simd128_int64.inl File Reference</a>	875
13.124.1	<a href="#">Macro Definition Documentation</a>	876
13.124.1.1	<a href="#">__FFLASFFPACK_fflas_ffpack_utils_simd128_int64_INL</a>	876
13.124.1.2	<a href="#">vect_t</a>	876
13.125	<a href="#">simd256.inl File Reference</a>	876
13.125.1	<a href="#">Macro Definition Documentation</a>	876
13.125.1.1	<a href="#">__FFLASFFPACK_fflas_ffpack_utils_simd256_INL</a>	876
13.125.2	<a href="#">Typedef Documentation</a>	876
13.125.2.1	<a href="#">Simd256</a>	876
13.126	<a href="#">simd256_double.inl File Reference</a>	876
13.126.1	<a href="#">Macro Definition Documentation</a>	877
13.126.1.1	<a href="#">__FFLASFFPACK_fflas_ffpack_utils_simd256_double_INL</a>	877
13.127	<a href="#">simd256_float.inl File Reference</a>	877
13.127.1	<a href="#">Macro Definition Documentation</a>	877
13.127.1.1	<a href="#">__FFLASFFPACK_fflas_ffpack_utils_simd256_float_INL</a>	877
13.128	<a href="#">simd256_int16.inl File Reference</a>	877
13.128.1	<a href="#">Macro Definition Documentation</a>	878

13.128.1.1 <a href="#">__FFLASFFPACK_fflas_ffpack_utils_simd256_int16_INL</a>	878
13.129 <a href="#">simd256_int32.inl</a> File Reference	878
13.129.1 Macro Definition Documentation	878
13.129.1.1 <a href="#">__FFLASFFPACK_fflas_ffpack_utils_simd256_int32_INL</a>	878
13.130 <a href="#">simd256_int64.inl</a> File Reference	878
13.130.1 Macro Definition Documentation	878
13.130.1.1 <a href="#">__FFLASFFPACK_fflas_ffpack_utils_simd256_int64_INL</a>	878
13.130.1.2 <a href="#">vect_t</a>	879
13.131 <a href="#">simd512.inl</a> File Reference	879
13.131.1 Macro Definition Documentation	879
13.131.1.1 <a href="#">__FFLASFFPACK_simd512_INL</a>	879
13.131.2 Typedef Documentation	879
13.131.2.1 <a href="#">Simd512</a>	879
13.132 <a href="#">simd512_double.inl</a> File Reference	879
13.132.1 Macro Definition Documentation	880
13.132.1.1 <a href="#">__FFLASFFPACK_simd512_double_INL</a>	880
13.133 <a href="#">simd512_float.inl</a> File Reference	880
13.133.1 Macro Definition Documentation	880
13.133.1.1 <a href="#">__FFLASFFPACK_simd512_float_INL</a>	880
13.134 <a href="#">simd512_int32.inl</a> File Reference	880
13.134.1 Macro Definition Documentation	880
13.134.1.1 <a href="#">__FFLASFFPACK_simd512_int32_INL</a>	880
13.135 <a href="#">simd512_int64.inl</a> File Reference	881
13.135.1 Macro Definition Documentation	881
13.135.1.1 <a href="#">_simd512_int64_INL</a>	881
13.135.1.2 <a href="#">vect_t</a>	881
13.136 <a href="#">simd_modular.inl</a> File Reference	881
13.137 <a href="#">fflas_sparse.h</a> File Reference	881
13.137.1 Macro Definition Documentation	885
13.137.1.1 <a href="#">index_t</a>	885
13.137.1.2 <a href="#">ROUND_DOWN</a>	885
13.137.1.3 <a href="#">__FFLASFFPACK_CACHE_LINE_SIZE</a>	885
13.137.1.4 <a href="#">assume_aligned</a>	885
13.137.1.5 <a href="#">DENSE_THRESHOLD</a>	885
13.138 <a href="#">fflas_sparse.inl</a> File Reference	886
13.138.1 Macro Definition Documentation	887
13.138.1.1 <a href="#">__FFLASFFPACK_fflas_fflas_sparse_INL</a>	887
13.139 <a href="#">coo.h</a> File Reference	888
13.140 <a href="#">coo_spm.inl</a> File Reference	888
13.140.1 Macro Definition Documentation	889
13.140.1.1 <a href="#">__FFLASFFPACK_fflas_sparse_coo_spm_INL</a>	889
13.141 <a href="#">coo_spmv.inl</a> File Reference	889

13.141.1 Macro Definition Documentation	890
13.141.1.1 __FFLASFFPACK_fflas_sparse_coo_spmv_INL	890
13.142 coo_utils.inl File Reference	890
13.142.1 Macro Definition Documentation	890
13.142.1.1 __FFLASFFPACK_fflas_sparse_coo_utils_INL	890
13.143 csr.h File Reference	890
13.144 csr_pspmm.inl File Reference	891
13.144.1 Macro Definition Documentation	892
13.144.1.1 __FFLASFFPACK_fflas_sparse_CSR_pspmm_INL	892
13.145 csr_pspmv.inl File Reference	892
13.145.1 Macro Definition Documentation	892
13.145.1.1 __FFLASFFPACK_fflas_sparse_CSR_pspmv_INL	892
13.146 csr_spm.inl File Reference	892
13.146.1 Macro Definition Documentation	893
13.146.1.1 __FFLASFFPACK_fflas_sparse_CSR_spm_INL	893
13.147 csr_spmv.inl File Reference	894
13.147.1 Macro Definition Documentation	894
13.147.1.1 __FFLASFFPACK_fflas_sparse_CSR_spmv_INL	894
13.148 csr_utils.inl File Reference	894
13.149 csr_hyb.h File Reference	895
13.150 csr_hyb_pspmm.inl File Reference	895
13.150.1 Macro Definition Documentation	896
13.150.1.1 __FFLASFFPACK_fflas_sparse_CSR_HYB_pspmm_INL	896
13.151 csr_hyb_pspmv.inl File Reference	896
13.151.1 Macro Definition Documentation	896
13.151.1.1 __FFLASFFPACK_fflas_sparse_CSR_HYB_pspmv_INL	896
13.152 csr_hyb_spm.inl File Reference	897
13.152.1 Macro Definition Documentation	897
13.152.1.1 __FFLASFFPACK_fflas_sparse_CSR_HYB_spm_INL	897
13.153 csr_hyb_spmv.inl File Reference	897
13.153.1 Macro Definition Documentation	897
13.153.1.1 __FFLASFFPACK_fflas_sparse_CSR_HYB_spmv_INL	897
13.154 csr_hyb_utils.inl File Reference	898
13.154.1 Macro Definition Documentation	898
13.154.1.1 __FFLASFFPACK_fflas_sparse_CSR_HYB_utils_INL	898
13.155 ell.h File Reference	898
13.156 ell_pspmm.inl File Reference	899
13.156.1 Macro Definition Documentation	899
13.156.1.1 __FFLASFFPACK_fflas_sparse_ELL_pspmm_INL	899
13.157 ell_pspmv.inl File Reference	899
13.157.1 Macro Definition Documentation	900
13.157.1.1 __FFLASFFPACK_fflas_sparse_ELL_pspmv_INL	900

13.158 ell_spm.inl File Reference . . . . .	900
13.158.1 Macro Definition Documentation . . . . .	901
13.158.1.1 __FflasFFPACK_fflas_sparse_ELL_spm_INL . . . . .	901
13.159 ell_spmv.inl File Reference . . . . .	901
13.159.1 Macro Definition Documentation . . . . .	902
13.159.1.1 __FflasFFPACK_fflas_sparse_ELL_spmv_INL . . . . .	902
13.160 ell_utils.inl File Reference . . . . .	902
13.160.1 Macro Definition Documentation . . . . .	902
13.160.1.1 __FflasFFPACK_fflas_sparse_ELL_utils_INL . . . . .	902
13.161 ell_simd.h File Reference . . . . .	902
13.162 ell_simd_pspmv.inl File Reference . . . . .	903
13.162.1 Macro Definition Documentation . . . . .	904
13.162.1.1 __FflasFFPACK_fflas_sparse_ELL_simd_pspmv_INL . . . . .	904
13.163 ell_simd_spmv.inl File Reference . . . . .	904
13.163.1 Macro Definition Documentation . . . . .	904
13.163.1.1 __FflasFFPACK_fflas_sparse_ELL_simd_spmv_INL . . . . .	904
13.164 ell_simd_utils.inl File Reference . . . . .	905
13.164.1 Macro Definition Documentation . . . . .	905
13.164.1.1 __FflasFFPACK_fflas_sparse_ELL_simd_utils_INL . . . . .	905
13.165 hyb_zo.h File Reference . . . . .	905
13.166 hyb_zo_pspmm.inl File Reference . . . . .	905
13.166.1 Macro Definition Documentation . . . . .	906
13.166.1.1 __FflasFFPACK_fflas_sparse_HYB_ZO_pspmm_INL . . . . .	906
13.167 hyb_zo_pspmv.inl File Reference . . . . .	906
13.167.1 Macro Definition Documentation . . . . .	906
13.167.1.1 __FflasFFPACK_fflas_sparse_HYB_ZO_pspmv_INL . . . . .	906
13.168 hyb_zo_spm.inl File Reference . . . . .	906
13.168.1 Macro Definition Documentation . . . . .	907
13.168.1.1 __FflasFFPACK_fflas_sparse_HYB_ZO_spm_INL . . . . .	907
13.169 hyb_zo_spmv.inl File Reference . . . . .	907
13.169.1 Macro Definition Documentation . . . . .	907
13.169.1.1 __FflasFFPACK_fflas_sparse_HYB_ZO_spmv_INL . . . . .	907
13.170 hyb_zo_utils.inl File Reference . . . . .	907
13.170.1 Macro Definition Documentation . . . . .	908
13.170.1.1 __FflasFFPACK_fflas_sparse_HYB_ZO_utils_INL . . . . .	908
13.171 read_sparse.h File Reference . . . . .	908
13.171.1 Macro Definition Documentation . . . . .	909
13.171.1.1 DNS_BIN_VER . . . . .	909
13.171.1.2 mask_t . . . . .	909
13.172 sell.h File Reference . . . . .	909
13.173 sell_pspmv.inl File Reference . . . . .	909
13.173.1 Macro Definition Documentation . . . . .	910

13.173.1.1 <a href="#">__FFLASFFPACK_fflas_sparse_sell_pspmv_INL</a> . . . . .	910
13.174 <a href="#">sell_spmv.inl</a> File Reference . . . . .	910
13.174.1 Macro Definition Documentation . . . . .	911
13.174.1.1 <a href="#">__FFLASFFPACK_fflas_sparse_sell_spmv_INL</a> . . . . .	911
13.175 <a href="#">sell_utils.inl</a> File Reference . . . . .	911
13.175.1 Macro Definition Documentation . . . . .	911
13.175.1.1 <a href="#">__FFLASFFPACK_fflas_sparse_sell_utils_INL</a> . . . . .	911
13.176 <a href="#">sparse_matrix_traits.h</a> File Reference . . . . .	912
13.177 <a href="#">utils.h</a> File Reference . . . . .	913
13.178 <a href="#">fflas_transpose.h</a> File Reference . . . . .	913
13.178.1 Detailed Description . . . . .	914
13.178.2 Macro Definition Documentation . . . . .	914
13.178.2.1 <a href="#">FFLAS_TRANSPOSE_BLOCKSIZE</a> . . . . .	914
13.178.2.2 LD . . . . .	914
13.178.2.3 ST . . . . .	914
13.179 <a href="#">ffpack.dox</a> File Reference . . . . .	914
13.180 <a href="#">ffpack.h</a> File Reference . . . . .	914
13.180.1 Detailed Description . . . . .	923
13.180.2 Macro Definition Documentation . . . . .	923
13.180.2.1 <a href="#">__FFLASFFPACK_FTRSTR_THRESHOLD</a> . . . . .	923
13.180.2.2 <a href="#">__FFLASFFPACK_FTRSSYR2K_THRESHOLD</a> . . . . .	923
13.181 <a href="#">ffpack.inl</a> File Reference . . . . .	924
13.181.1 Macro Definition Documentation . . . . .	925
13.181.1.1 <a href="#">__FFLASFFPACK_ffpack_INL</a> . . . . .	925
13.182 <a href="#">ffpack_bruhatgen.inl</a> File Reference . . . . .	925
13.182.1 Macro Definition Documentation . . . . .	926
13.182.1.1 <a href="#">__FFLASFFPACK_ffpack_bruhatgen_inl</a> . . . . .	926
13.183 <a href="#">ffpack_charpoly.inl</a> File Reference . . . . .	926
13.183.1 Macro Definition Documentation . . . . .	927
13.183.1.1 <a href="#">__FFLASFFPACK_charpoly_INL</a> . . . . .	927
13.184 <a href="#">ffpack_charpoly_danilevski.inl</a> File Reference . . . . .	927
13.184.1 Macro Definition Documentation . . . . .	927
13.184.1.1 <a href="#">__FFLASFFPACK_ffpack_charpoly_danilveski_INL</a> . . . . .	927
13.185 <a href="#">ffpack_charpoly_kgfast.inl</a> File Reference . . . . .	927
13.185.1 Macro Definition Documentation . . . . .	928
13.185.1.1 <a href="#">__FFLASFFPACK_ffpack_charpoly_kgfast_INL</a> . . . . .	928
13.186 <a href="#">ffpack_charpoly_kgfastgeneralized.inl</a> File Reference . . . . .	928
13.186.1 Macro Definition Documentation . . . . .	928
13.186.1.1 <a href="#">__FFLASFFPACK_ffpack_charpoly_kgfastgeneralized_INL</a> . . . . .	928
13.187 <a href="#">ffpack_charpoly_kglu.inl</a> File Reference . . . . .	928
13.187.1 Macro Definition Documentation . . . . .	929
13.187.1.1 <a href="#">__FFLASFFPACK_ffpack_charpoly_kglu_INL</a> . . . . .	929



13.188 fpack_charpoly_mp.inl File Reference . . . . .	929
13.188.1 Macro Definition Documentation . . . . .	929
13.188.1.1 __FFPACK_charpoly_mp_INL . . . . .	929
13.189 fpack_det_mp.inl File Reference . . . . .	929
13.189.1 Macro Definition Documentation . . . . .	930
13.189.1.1 __FFPACK_det_mp_INL . . . . .	930
13.190 fpack_echelonforms.inl File Reference . . . . .	930
13.190.1 Macro Definition Documentation . . . . .	931
13.190.1.1 __FFLASFFPACK_fpack_echelon_forms_INL . . . . .	931
13.190.1.2 __FFLASFFPACK_GAUSSJORDAN_BASECASE . . . . .	931
13.191 fpack_fgesv.inl File Reference . . . . .	931
13.191.1 Macro Definition Documentation . . . . .	932
13.191.1.1 __FFLASFFPACK_fpack_fgesv_INL . . . . .	932
13.192 fpack_fgetrs.inl File Reference . . . . .	932
13.192.1 Macro Definition Documentation . . . . .	932
13.192.1.1 __FFLASFFPACK_fpack_fgetrs_INL . . . . .	932
13.193 fpack_frobenius.inl File Reference . . . . .	932
13.194 fpack_fsytrf.inl File Reference . . . . .	933
13.194.1 Macro Definition Documentation . . . . .	934
13.194.1.1 __FFLASFFPACK_fpack_fsytrf_INL . . . . .	934
13.195 fpack_ftrssyr2k.inl File Reference . . . . .	934
13.195.1 Macro Definition Documentation . . . . .	935
13.195.1.1 __FFLASFFPACK_fpack_ftrssyr2k_INL . . . . .	935
13.196 fpack_ftrstr.inl File Reference . . . . .	935
13.196.1 Macro Definition Documentation . . . . .	935
13.196.1.1 __FFLASFFPACK_fpack_ftrstr_INL . . . . .	935
13.197 fpack_ftrtr.inl File Reference . . . . .	935
13.197.1 Macro Definition Documentation . . . . .	936
13.197.1.1 ENABLE_ALL_CHECKINGS . . . . .	936
13.197.1.2 __FFLASFFPACK_fpack_ftrtr_INL . . . . .	936
13.198 fpack_invert.inl File Reference . . . . .	936
13.198.1 Macro Definition Documentation . . . . .	936
13.198.1.1 __FFLASFFPACK_fpack_invert_INL . . . . .	936
13.199 fpack_krylovelim.inl File Reference . . . . .	937
13.199.1 Macro Definition Documentation . . . . .	937
13.199.1.1 __FFLASFFPACK_fpack_krylovelim_INL . . . . .	937
13.200 fpack_ludivine.inl File Reference . . . . .	937
13.200.1 Macro Definition Documentation . . . . .	937
13.200.1.1 __FFLASFFPACK_fpack_ludivine_INL . . . . .	937
13.201 fpack_ludivine_mp.inl File Reference . . . . .	938
13.201.1 Macro Definition Documentation . . . . .	938
13.201.1.1 __FFPACK_ludivine_mp_INL . . . . .	938

13.202 fpack_minpoly.inl File Reference . . . . .	938
13.202.1 Macro Definition Documentation . . . . .	939
13.202.1.1 __FFLASFFPACK_fpack_minpoly_INL . . . . .	939
13.203 fpack_permutation.inl File Reference . . . . .	939
13.203.1 Macro Definition Documentation . . . . .	941
13.203.1.1 __FFLASFFPACK_fpack_permutation_INL . . . . .	941
13.203.1.2 FFLASFFPACK_PERM_BKSIZE . . . . .	941
13.204 fpack_pluq.inl File Reference . . . . .	941
13.204.1 Macro Definition Documentation . . . . .	942
13.204.1.1 __FFLASFFPACK_fpack_pluq_INL . . . . .	942
13.204.1.2 CROUT . . . . .	942
13.205 fpack_pluq_mp.inl File Reference . . . . .	942
13.205.1 Macro Definition Documentation . . . . .	942
13.205.1.1 __FFPACK_pluq_mp_INL . . . . .	942
13.206 fpack_ppluq.inl File Reference . . . . .	942
13.206.1 Macro Definition Documentation . . . . .	943
13.206.1.1 __FFLASFFPACK_fpack_ppluq_INL . . . . .	943
13.206.1.2 __FFLAS__TRSM_READONLY . . . . .	943
13.206.1.3 PBASECASE_K . . . . .	943
13.207 fpack_rankprofiles.inl File Reference . . . . .	943
13.207.1 Macro Definition Documentation . . . . .	944
13.207.1.1 __FFLASFFPACK_fpack_rank_profiles_INL . . . . .	944
13.208 field-traits.h File Reference . . . . .	944
13.208.1 Detailed Description . . . . .	947
13.209 field.doxy File Reference . . . . .	947
13.210 rns-double-elt.h File Reference . . . . .	947
13.210.1 Detailed Description . . . . .	947
13.211 rns-double-recint.inl File Reference . . . . .	947
13.211.1 Macro Definition Documentation . . . . .	948
13.211.1.1 __FFLASFFPACK_field_rns_double_recint_INL . . . . .	948
13.212 rns-double.h File Reference . . . . .	948
13.212.1 Detailed Description . . . . .	948
13.212.2 Macro Definition Documentation . . . . .	948
13.212.2.1 ROUND_DOWN . . . . .	948
13.213 rns-double.inl File Reference . . . . .	949
13.213.1 Macro Definition Documentation . . . . .	949
13.213.1.1 __FFLASFFPACK_field_rns_double_INL . . . . .	949
13.214 rns-integer-mod.h File Reference . . . . .	949
13.214.1 Detailed Description . . . . .	950
13.215 rns-integer.h File Reference . . . . .	950
13.215.1 Detailed Description . . . . .	950
13.216 rns.h File Reference . . . . .	950

13.217 rns.inl File Reference	951
13.217.1 Macro Definition Documentation	951
13.217.1.1 __FFLASFFPACK_field_rns_INL	951
13.218 interfaces.doxy File Reference	951
13.219 fflas_c.h File Reference	951
13.219.1 Macro Definition Documentation	953
13.219.1.1 FFLAS_COMPILED	953
13.219.2 Enumeration Type Documentation	953
13.219.2.1 FFLAS_C_ORDER	953
13.219.2.2 FFLAS_C_TRANSPOSE	953
13.219.2.3 FFLAS_C_UPLO	953
13.219.2.4 FFLAS_C_DIAG	954
13.219.2.5 FFLAS_C_SIDE	954
13.219.2.6 FFLAS_C_BASE	954
13.219.3 Function Documentation	954
13.219.3.1 freducein_1_modular_double()	954
13.219.3.2 freduce_1_modular_double()	955
13.219.3.3 fnegin_1_modular_double()	955
13.219.3.4 fneg_1_modular_double()	955
13.219.3.5 fzero_1_modular_double()	955
13.219.3.6 fiszero_1_modular_double()	955
13.219.3.7 fequal_1_modular_double()	955
13.219.3.8 fassign_1_modular_double()	956
13.219.3.9 fscaln_1_modular_double()	956
13.219.3.10 fscal_1_modular_double()	956
13.219.3.11 faxpy_1_modular_double()	956
13.219.3.12 fdot_1_modular_double()	956
13.219.3.13 fswap_1_modular_double()	956
13.219.3.14 fadd_1_modular_double()	957
13.219.3.15 fsub_1_modular_double()	957
13.219.3.16 faddin_1_modular_double()	957
13.219.3.17 fsubin_1_modular_double()	957
13.219.3.18 fassign_2_modular_double()	957
13.219.3.19 fzero_2_modular_double()	958
13.219.3.20 fequal_2_modular_double()	958
13.219.3.21 fiszero_2_modular_double()	958
13.219.3.22 fidentity_2_modular_double()	958
13.219.3.23 freducein_2_modular_double()	958
13.219.3.24 freduce_2_modular_double()	959
13.219.3.25 fnegin_2_modular_double()	959
13.219.3.26 fneg_2_modular_double()	959
13.219.3.27 fscaln_2_modular_double()	959

13.219.3.28 fscal_2_modular_double()	959
13.219.3.29 faxpy_2_modular_double()	960
13.219.3.30 fmove_2_modular_double()	960
13.219.3.31 fadd_2_modular_double()	960
13.219.3.32 fsub_2_modular_double()	960
13.219.3.33 fsubin_2_modular_double()	960
13.219.3.34 faddin_2_modular_double()	961
13.219.3.35 fgemv_2_modular_double()	961
13.219.3.36 fger_2_modular_double()	961
13.219.3.37 ftrsv_2_modular_double()	961
13.219.3.38 ftrsm_3_modular_double()	962
13.219.3.39 ftrmm_3_modular_double()	962
13.219.3.40 fgemm_3_modular_double()	962
13.219.3.41 fsquare_3_modular_double()	963
13.220 fflas_L1_inst.C File Reference	963
13.220.1 Macro Definition Documentation	963
13.220.1.1 __FFLAS_L1_INST_C	963
13.220.1.2 INST_OR_DECL	963
13.220.1.3 FFLAS_FIELD [1/2]	963
13.220.1.4 FFLAS_ELT [1/6]	963
13.220.1.5 FFLAS_ELT [2/6]	964
13.220.1.6 FFLAS_ELT [3/6]	964
13.220.1.7 FFLAS_FIELD [2/2]	964
13.220.1.8 FFLAS_ELT [4/6]	964
13.220.1.9 FFLAS_ELT [5/6]	964
13.220.1.10 FFLAS_ELT [6/6]	964
13.221 fflas_L1_inst.h File Reference	964
13.221.1 Macro Definition Documentation	964
13.221.1.1 INST_OR_DECL	964
13.221.1.2 FFLAS_FIELD [1/2]	964
13.221.1.3 FFLAS_ELT [1/6]	964
13.221.1.4 FFLAS_ELT [2/6]	965
13.221.1.5 FFLAS_ELT [3/6]	965
13.221.1.6 FFLAS_FIELD [2/2]	965
13.221.1.7 FFLAS_ELT [4/6]	965
13.221.1.8 FFLAS_ELT [5/6]	965
13.221.1.9 FFLAS_ELT [6/6]	965
13.222 fflas_L1_inst_implem.inl File Reference	965
13.223 fflas_L2_inst.C File Reference	966
13.223.1 Macro Definition Documentation	967
13.223.1.1 __FFLAS_L2_INST_C	967
13.223.1.2 INST_OR_DECL	967

13.223.1.3 FFLAS_FIELD [1/2]	967
13.223.1.4 FFLAS_ELT [1/6]	967
13.223.1.5 FFLAS_ELT [2/6]	967
13.223.1.6 FFLAS_ELT [3/6]	967
13.223.1.7 FFLAS_FIELD [2/2]	967
13.223.1.8 FFLAS_ELT [4/6]	967
13.223.1.9 FFLAS_ELT [5/6]	967
13.223.1.10 FFLAS_ELT [6/6]	967
13.224 fflas_L2_inst.h File Reference	967
13.224.1 Macro Definition Documentation	968
13.224.1.1 INST_OR_DECL	968
13.224.1.2 FFLAS_FIELD [1/2]	968
13.224.1.3 FFLAS_ELT [1/6]	968
13.224.1.4 FFLAS_ELT [2/6]	968
13.224.1.5 FFLAS_ELT [3/6]	968
13.224.1.6 FFLAS_FIELD [2/2]	968
13.224.1.7 FFLAS_ELT [4/6]	968
13.224.1.8 FFLAS_ELT [5/6]	968
13.224.1.9 FFLAS_ELT [6/6]	968
13.225 fflas_L2_inst_implem.inl File Reference	968
13.226 fflas_L3_inst.C File Reference	970
13.226.1 Macro Definition Documentation	970
13.226.1.1 __FFLAS_L3_INST_C	970
13.226.1.2 INST_OR_DECL	970
13.226.1.3 FFLAS_FIELD [1/2]	970
13.226.1.4 FFLAS_ELT [1/6]	970
13.226.1.5 FFLAS_ELT [2/6]	971
13.226.1.6 FFLAS_ELT [3/6]	971
13.226.1.7 FFLAS_FIELD [2/2]	971
13.226.1.8 FFLAS_ELT [4/6]	971
13.226.1.9 FFLAS_ELT [5/6]	971
13.226.1.10 FFLAS_ELT [6/6]	971
13.227 fflas_L3_inst.h File Reference	971
13.227.1 Macro Definition Documentation	971
13.227.1.1 INST_OR_DECL	971
13.227.1.2 FFLAS_FIELD [1/2]	971
13.227.1.3 FFLAS_ELT [1/6]	971
13.227.1.4 FFLAS_ELT [2/6]	972
13.227.1.5 FFLAS_ELT [3/6]	972
13.227.1.6 FFLAS_FIELD [2/2]	972
13.227.1.7 FFLAS_ELT [4/6]	972
13.227.1.8 FFLAS_ELT [5/6]	972

13.227.1.9 FFLAS_ELT [6/6]	972
13.228 fflas_L3_inst_implem.inl File Reference	972
13.228.1 Macro Definition Documentation	973
13.228.1.1 __FFLAS__TRSM_READONLY	973
13.229 fflas_lvl1.C File Reference	973
13.229.1 Detailed Description	974
13.229.2 Function Documentation	974
13.229.2.1 freducein_1_modular_double()	974
13.229.2.2 freduce_1_modular_double()	974
13.229.2.3 fnegin_1_modular_double()	974
13.229.2.4 fneg_1_modular_double()	974
13.229.2.5 fzero_1_modular_double()	975
13.229.2.6 fiszero_1_modular_double()	975
13.229.2.7 fequal_1_modular_double()	975
13.229.2.8 fassign_1_modular_double()	975
13.229.2.9 fscaln_1_modular_double()	975
13.229.2.10 fscal_1_modular_double()	975
13.229.2.11 faxpy_1_modular_double()	976
13.229.2.12 fdot_1_modular_double()	976
13.229.2.13 fswap_1_modular_double()	976
13.229.2.14 fadd_1_modular_double()	976
13.229.2.15 fsub_1_modular_double()	976
13.229.2.16 faddn_1_modular_double()	977
13.229.2.17 fsubn_1_modular_double()	977
13.230 fflas_lvl2.C File Reference	977
13.230.1 Detailed Description	978
13.230.2 Function Documentation	978
13.230.2.1 fassign_2_modular_double()	978
13.230.2.2 fzero_2_modular_double()	978
13.230.2.3 fequal_2_modular_double()	979
13.230.2.4 fiszero_2_modular_double()	979
13.230.2.5 fidentity_2_modular_double()	979
13.230.2.6 freducein_2_modular_double()	979
13.230.2.7 freduce_2_modular_double()	979
13.230.2.8 fnegin_2_modular_double()	979
13.230.2.9 fneg_2_modular_double()	980
13.230.2.10 fscaln_2_modular_double()	980
13.230.2.11 fscal_2_modular_double()	980
13.230.2.12 faxpy_2_modular_double()	980
13.230.2.13 fmove_2_modular_double()	980
13.230.2.14 fadd_2_modular_double()	981
13.230.2.15 fsub_2_modular_double()	981

13.230.2.16	<a href="#">fsubin_2_modular_double()</a>	981
13.230.2.17	<a href="#">faddin_2_modular_double()</a>	981
13.230.2.18	<a href="#">fgemv_2_modular_double()</a>	982
13.230.2.19	<a href="#">fger_2_modular_double()</a>	982
13.230.2.20	<a href="#">ftrsv_2_modular_double()</a>	982
13.231	<a href="#">fflas_lvl3.C File Reference</a>	982
13.231.1	<a href="#">Detailed Description</a>	983
13.231.2	<a href="#">Function Documentation</a>	983
13.231.2.1	<a href="#">ftrsm_3_modular_double()</a>	983
13.231.2.2	<a href="#">ftrmm_3_modular_double()</a>	983
13.231.2.3	<a href="#">fgemm_3_modular_double()</a>	984
13.231.2.4	<a href="#">fsquare_3_modular_double()</a>	984
13.232	<a href="#">fflas_sparse.C File Reference</a>	984
13.232.1	<a href="#">Detailed Description</a>	984
13.233	<a href="#">ffpack.C File Reference</a>	984
13.233.1	<a href="#">Detailed Description</a>	988
13.233.2	<a href="#">Function Documentation</a>	988
13.233.2.1	<a href="#">LAPACKPerm2MathPerm()</a>	988
13.233.2.2	<a href="#">MathPerm2LAPACKPerm()</a>	988
13.233.2.3	<a href="#">MatrixApplyS_modular_double()</a>	988
13.233.2.4	<a href="#">PermApplyS_double()</a>	988
13.233.2.5	<a href="#">MatrixApplyT_modular_double()</a>	989
13.233.2.6	<a href="#">PermApplyT_double()</a>	989
13.233.2.7	<a href="#">composePermutationsLLM()</a>	989
13.233.2.8	<a href="#">composePermutationsLLL()</a>	989
13.233.2.9	<a href="#">composePermutationsMLM()</a>	989
13.233.2.10	<a href="#">cyclic_shift_mathPerm()</a>	989
13.233.2.11	<a href="#">cyclic_shift_row_modular_double()</a>	990
13.233.2.12	<a href="#">cyclic_shift_col_modular_double()</a>	990
13.233.2.13	<a href="#">applyP_modular_double()</a>	990
13.233.2.14	<a href="#">fgetrsin_modular_double()</a>	990
13.233.2.15	<a href="#">fgetrsv_modular_double()</a>	990
13.233.2.16	<a href="#">fgesvin_modular_double()</a>	991
13.233.2.17	<a href="#">fgesv_modular_double()</a>	991
13.233.2.18	<a href="#">ftrtri_modular_double()</a>	991
13.233.2.19	<a href="#">trinv_left_modular_double()</a>	991
13.233.2.20	<a href="#">ftrtm_modular_double()</a>	992
13.233.2.21	<a href="#">PLUQ_modular_double()</a>	992
13.233.2.22	<a href="#">LUdivine_modular_double()</a>	992
13.233.2.23	<a href="#">ColumnEchelonForm_modular_double()</a>	992
13.233.2.24	<a href="#">RowEchelonForm_modular_double()</a>	993
13.233.2.25	<a href="#">ReducedColumnEchelonForm_modular_double()</a>	993

13.233.2.26 ReducedRowEchelonForm_modular_double()	993
13.233.2.27 ColumnEchelonForm_modular_float()	993
13.233.2.28 RowEchelonForm_modular_float()	993
13.233.2.29 ReducedColumnEchelonForm_modular_float()	994
13.233.2.30 ReducedRowEchelonForm_modular_float()	994
13.233.2.31 ColumnEchelonForm_modular_int32_t()	994
13.233.2.32 RowEchelonForm_modular_int32_t()	994
13.233.2.33 ReducedColumnEchelonForm_modular_int32_t()	995
13.233.2.34 ReducedRowEchelonForm_modular_int32_t()	995
13.233.2.35 pColumnEchelonForm_modular_double()	995
13.233.2.36 pRowEchelonForm_modular_double()	995
13.233.2.37 pReducedColumnEchelonForm_modular_double()	996
13.233.2.38 pReducedRowEchelonForm_modular_double()	996
13.233.2.39 pColumnEchelonForm_modular_float()	996
13.233.2.40 pRowEchelonForm_modular_float()	996
13.233.2.41 pReducedColumnEchelonForm_modular_float()	997
13.233.2.42 pReducedRowEchelonForm_modular_float()	997
13.233.2.43 pColumnEchelonForm_modular_int32_t()	997
13.233.2.44 pRowEchelonForm_modular_int32_t()	997
13.233.2.45 pReducedColumnEchelonForm_modular_int32_t()	997
13.233.2.46 pReducedRowEchelonForm_modular_int32_t()	998
13.233.2.47 Invertin_modular_double()	998
13.233.2.48 Invert_modular_double()	998
13.233.2.49 Invert2_modular_double()	998
13.233.2.50 KrylovElim_modular_double()	999
13.233.2.51 SpecRankProfile_modular_double()	999
13.233.2.52 Rank_modular_double()	999
13.233.2.53 IsSingular_modular_double()	999
13.233.2.54 Det_modular_double()	999
13.233.2.55 Solve_modular_double()	1000
13.233.2.56 solveLB_modular_double()	1000
13.233.2.57 solveLB2_modular_double()	1000
13.233.2.58 RandomNullSpaceVector_modular_double()	1000
13.233.2.59 NullSpaceBasis_modular_double()	1000
13.233.2.60 RowRankProfile_modular_double()	1001
13.233.2.61 ColumnRankProfile_modular_double()	1001
13.233.2.62 RankProfileFromLU()	1001
13.233.2.63 LeadingSubmatrixRankProfiles()	1001
13.233.2.64 RowRankProfileSubmatrixIndices_modular_double()	1001
13.233.2.65 ColRankProfileSubmatrixIndices_modular_double()	1002
13.233.2.66 RowRankProfileSubmatrix_modular_double()	1002
13.233.2.67 ColRankProfileSubmatrix_modular_double()	1002



13.233.2.68	getTriangular_modular_double()	1002
13.233.2.69	getTriangularin_modular_double()	1003
13.233.2.70	getEchelonForm_modular_double()	1003
13.233.2.71	getEchelonFormin_modular_double()	1003
13.233.2.72	getEchelonTransform_modular_double()	1003
13.233.2.73	getReducedEchelonForm_modular_double()	1004
13.233.2.74	getReducedEchelonFormin_modular_double()	1004
13.233.2.75	getReducedEchelonTransform_modular_double()	1004
13.233.2.76	PLUQtoEchelonPermutation()	1005
13.234	ffpack_c.h File Reference	1005
13.234.1	Macro Definition Documentation	1008
13.234.1.1	FFPACK_COMPILED	1008
13.234.2	Enumeration Type Documentation	1008
13.234.2.1	FFLAS_C_ORDER	1008
13.234.2.2	FFLAS_C_TRANSPOSE	1008
13.234.2.3	FFLAS_C_UPLO	1008
13.234.2.4	FFLAS_C_DIAG	1008
13.234.2.5	FFLAS_C_SIDE	1009
13.234.2.6	FFPACK_C_LU_TAG	1009
13.234.2.7	FFPACK_C_CHARPOLY_TAG	1009
13.234.2.8	FFPACK_C_MINPOLY_TAG	1009
13.234.3	Function Documentation	1009
13.234.3.1	LAPACKPerm2MathPerm()	1009
13.234.3.2	MathPerm2LAPACKPerm()	1010
13.234.3.3	MatrixApplyS_modular_double()	1010
13.234.3.4	PermApplyS_double()	1010
13.234.3.5	MatrixApplyT_modular_double()	1010
13.234.3.6	PermApplyT_double()	1010
13.234.3.7	composePermutationsLLM()	1011
13.234.3.8	composePermutationsLLL()	1011
13.234.3.9	composePermutationsMLM()	1011
13.234.3.10	cyclic_shift_mathPerm()	1011
13.234.3.11	cyclic_shift_row_modular_double()	1011
13.234.3.12	cyclic_shift_col_modular_double()	1011
13.234.3.13	applyP_modular_double()	1011
13.234.3.14	fgetrsin_modular_double()	1012
13.234.3.15	fgetrs_modular_double()	1012
13.234.3.16	fgesvin_modular_double()	1012
13.234.3.17	fgesv_modular_double()	1012
13.234.3.18	fttrtri_modular_double()	1013
13.234.3.19	trinv_left_modular_double()	1013
13.234.3.20	fttrtm_modular_double()	1013

13.234.3.21 PLUQ_modular_double()	1013
13.234.3.22 LUdivine_modular_double()	1014
13.234.3.23 LUdivine_small_modular_double()	1014
13.234.3.24 LUdivine_gauss_modular_double()	1014
13.234.3.25 ColumnEchelonForm_modular_double()	1014
13.234.3.26 RowEchelonForm_modular_double()	1015
13.234.3.27 ColumnEchelonForm_modular_float()	1015
13.234.3.28 RowEchelonForm_modular_float()	1015
13.234.3.29 ColumnEchelonForm_modular_int32_t()	1015
13.234.3.30 RowEchelonForm_modular_int32_t()	1015
13.234.3.31 ReducedColumnEchelonForm_modular_double()	1016
13.234.3.32 ReducedRowEchelonForm_modular_double()	1016
13.234.3.33 ReducedColumnEchelonForm_modular_float()	1016
13.234.3.34 ReducedRowEchelonForm_modular_float()	1016
13.234.3.35 ReducedColumnEchelonForm_modular_int32_t()	1017
13.234.3.36 ReducedRowEchelonForm_modular_int32_t()	1017
13.234.3.37 ReducedRowEchelonForm2_modular_double()	1017
13.234.3.38 REF_modular_double()	1017
13.234.3.39 Invertin_modular_double()	1018
13.234.3.40 Invert_modular_double()	1018
13.234.3.41 Invert2_modular_double()	1018
13.234.3.42 KrylovElim_modular_double()	1018
13.234.3.43 SpecRankProfile_modular_double()	1018
13.234.3.44 Rank_modular_double()	1019
13.234.3.45 IsSingular_modular_double()	1019
13.234.3.46 Det_modular_double()	1019
13.234.3.47 Solve_modular_double()	1019
13.234.3.48 solveLB_modular_double()	1019
13.234.3.49 solveLB2_modular_double()	1020
13.234.3.50 RandomNullSpaceVector_modular_double()	1020
13.234.3.51 NullSpaceBasis_modular_double()	1020
13.234.3.52 RowRankProfile_modular_double()	1020
13.234.3.53 ColumnRankProfile_modular_double()	1021
13.234.3.54 RankProfileFromLU()	1021
13.234.3.55 LeadingSubmatrixRankProfiles()	1021
13.234.3.56 RowRankProfileSubmatrixIndices_modular_double()	1021
13.234.3.57 ColRankProfileSubmatrixIndices_modular_double()	1021
13.234.3.58 RowRankProfileSubmatrix_modular_double()	1022
13.234.3.59 ColRankProfileSubmatrix_modular_double()	1022
13.234.3.60 getTriangular_modular_double()	1022
13.234.3.61 getTriangularin_modular_double()	1022
13.234.3.62 getEchelonForm_modular_double()	1022

13.234.3.63 getEchelonFormin_modular_double()	1023
13.234.3.64 getEchelonTransform_modular_double()	1023
13.234.3.65 getReducedEchelonForm_modular_double()	1023
13.234.3.66 getReducedEchelonFormin_modular_double()	1024
13.234.3.67 getReducedEchelonTransform_modular_double()	1024
13.234.3.68 PLUQtoEchelonPermutation()	1024
13.235 fpack_inst.C File Reference	1024
13.235.1 Macro Definition Documentation	1025
13.235.1.1 __FFPACK_INST_C	1025
13.235.1.2 FFLAS_COMPILED	1025
13.235.1.3 INST_OR_DECL	1025
13.235.1.4 FFLAS_FIELD [1/2]	1025
13.235.1.5 FFLAS_ELT [1/6]	1025
13.235.1.6 FFLAS_ELT [2/6]	1025
13.235.1.7 FFLAS_ELT [3/6]	1025
13.235.1.8 FFLAS_FIELD [2/2]	1025
13.235.1.9 FFLAS_ELT [4/6]	1025
13.235.1.10 FFLAS_ELT [5/6]	1025
13.235.1.11 FFLAS_ELT [6/6]	1025
13.236 fpack_inst.h File Reference	1025
13.236.1 Macro Definition Documentation	1026
13.236.1.1 FFLAS_COMPILED	1026
13.236.1.2 INST_OR_DECL	1026
13.236.1.3 FFLAS_FIELD [1/2]	1026
13.236.1.4 FFLAS_ELT [1/6]	1026
13.236.1.5 FFLAS_ELT [2/6]	1026
13.236.1.6 FFLAS_ELT [3/6]	1026
13.236.1.7 FFLAS_FIELD [2/2]	1026
13.236.1.8 FFLAS_ELT [4/6]	1026
13.236.1.9 FFLAS_ELT [5/6]	1026
13.236.1.10 FFLAS_ELT [6/6]	1026
13.237 fpack_inst_implem.inl File Reference	1027
13.238 blockcuts.inl File Reference	1030
13.238.1 Macro Definition Documentation	1031
13.238.1.1 __FFLASFFPACK_fflas_blockcuts_INL	1031
13.238.1.2 __FFLASFFPACK_MINBLOCKCUTS	1031
13.239 fflas_plevel1.h File Reference	1031
13.240 kaapi_routines.inl File Reference	1032
13.240.1 Macro Definition Documentation	1032
13.240.1.1 __FFLASFFPACK_KAAPI_ROUTINES_INL	1032
13.241 parallel.h File Reference	1032
13.241.1 Macro Definition Documentation	1033

13.241.1.1 __FFLASFFPACK_SEQUENTIAL . . . . .	1033
13.241.1.2 index_t . . . . .	1033
13.241.1.3 TASK . . . . .	1033
13.241.1.4 WAIT . . . . .	1033
13.241.1.5 CHECK_DEPENDENCIES . . . . .	1033
13.241.1.6 BARRIER . . . . .	1033
13.241.1.7 PAR_BLOCK . . . . .	1033
13.241.1.8 SYNCH_GROUP . . . . .	1033
13.241.1.9 THREAD_INDEX . . . . .	1034
13.241.1.10 NUM_THREADS . . . . .	1034
13.241.1.11 SET_THREADS . . . . .	1034
13.241.1.12 MAX_THREADS . . . . .	1034
13.241.1.13 READ . . . . .	1034
13.241.1.14 WRITE . . . . .	1034
13.241.1.15 READWRITE . . . . .	1034
13.241.1.16 CONSTREFERENCE . . . . .	1034
13.241.1.17 VALUE . . . . .	1034
13.241.1.18 BEGIN_PARALLEL_MAIN . . . . .	1034
13.241.1.19 END_PARALLEL_MAIN . . . . .	1034
13.241.1.20 FORBLOCK1D . . . . .	1035
13.241.1.21 FOR1D . . . . .	1035
13.241.1.22 PARFORBLOCK1D . . . . .	1035
13.241.1.23 PARFOR1D . . . . .	1035
13.241.1.24 FORBLOCK2D . . . . .	1035
13.241.1.25 FOR2D . . . . .	1035
13.241.1.26 PARFORBLOCK2D . . . . .	1036
13.241.1.27 PARFOR2D . . . . .	1036
13.241.1.28 COMMA . . . . .	1036
13.241.1.29 MODE . . . . .	1036
13.241.1.30 RETURNPARAM . . . . .	1036
13.241.1.31 NUMARGS . . . . .	1036
13.241.1.32 PP_NARG_ . . . . .	1037
13.241.1.33 PP_ARG_N . . . . .	1037
13.241.1.34 PP_RSEQ_N . . . . .	1038
13.241.1.35 NOSPLIT . . . . .	1038
13.241.1.36 splitting_0 . . . . .	1038
13.241.1.37 splitting_1 . . . . .	1038
13.241.1.38 splitting_2 . . . . .	1038
13.241.1.39 splitting_3 . . . . .	1038
13.241.1.40 splitt . . . . .	1039
13.241.1.41 SPLITTER . . . . .	1039
13.242 pfgemm_variants.inl File Reference . . . . .	1039

13.243 pfgemv.ini File Reference . . . . .	1040
13.244 align-allocator.h File Reference . . . . .	1040
13.245 args-parser.h File Reference . . . . .	1040
13.245.1 Macro Definition Documentation . . . . .	1041
13.245.1.1 TYPE_BOOL . . . . .	1041
13.245.1.2 END_OF_ARGUMENTS . . . . .	1041
13.245.1.3 type_integer . . . . .	1041
13.245.2 Enumeration Type Documentation . . . . .	1041
13.245.2.1 ArgumentType . . . . .	1041
13.245.3 Function Documentation . . . . .	1042
13.245.3.1 printHelpMessage() . . . . .	1042
13.245.3.2 findArgument() . . . . .	1042
13.245.3.3 getListArgs() . . . . .	1042
13.246 bit_manipulation.h File Reference . . . . .	1042
13.246.1 Macro Definition Documentation . . . . .	1042
13.246.1.1 __has_builtin . . . . .	1042
13.246.2 Function Documentation . . . . .	1043
13.246.2.1 clz() [1/2] . . . . .	1043
13.246.2.2 clz() [2/2] . . . . .	1043
13.246.2.3 ctz() [1/2] . . . . .	1043
13.246.2.4 ctz() [2/2] . . . . .	1043
13.247 cast.h File Reference . . . . .	1043
13.248 debug.h File Reference . . . . .	1043
13.248.1 Detailed Description . . . . .	1044
13.248.2 Macro Definition Documentation . . . . .	1044
13.248.2.1 FFLASFFPACK_check . . . . .	1044
13.248.2.2 FFLASFFPACK_abort . . . . .	1044
13.249 fflas_intrinsic.h File Reference . . . . .	1044
13.250 fflas_io.h File Reference . . . . .	1044
13.251 fflas_memory.h File Reference . . . . .	1045
13.252 fflas_randommatrix.h File Reference . . . . .	1046
13.253 flimits.h File Reference . . . . .	1048
13.253.1 Function Documentation . . . . .	1048
13.253.1.1 in_range() [1/3] . . . . .	1048
13.253.1.2 in_range() [2/3] . . . . .	1049
13.253.1.3 in_range() [3/3] . . . . .	1049
13.254 Matio.h File Reference . . . . .	1049
13.254.1 Function Documentation . . . . .	1049
13.254.1.1 read_field() . . . . .	1049
13.254.1.2 write_field() . . . . .	1049
13.255 test-utils.h File Reference . . . . .	1049
13.256 timer.h File Reference . . . . .	1050

13.257 cblas.C File Reference	1050
13.257.1 Macro Definition Documentation	1051
13.257.1.1 __FFLASFFPACK_CONFIGURATION	1051
13.257.1.2 __FFLASFFPACK_HAVE_CBLAS	1051
13.257.2 Function Documentation	1051
13.257.2.1 main()	1051
13.258 clapack.C File Reference	1051
13.258.1 Macro Definition Documentation	1051
13.258.1.1 __FFLASFFPACK_CONFIGURATION	1051
13.258.1.2 __FFLASFFPACK_HAVE_LAPACK	1051
13.258.1.3 __FFLASFFPACK_HAVE_CLAPACK	1051
13.258.2 Function Documentation	1051
13.258.2.1 main()	1051
13.259 cuda.C File Reference	1052
13.259.1 Function Documentation	1052
13.259.1.1 main()	1052
13.260 fblas.C File Reference	1052
13.260.1 Macro Definition Documentation	1052
13.260.1.1 __FFLASFFPACK_CONFIGURATION	1052
13.260.2 Function Documentation	1052
13.260.2.1 dgemm_()	1052
13.260.2.2 main()	1053
13.261 lapack.C File Reference	1053
13.261.1 Macro Definition Documentation	1053
13.261.1.1 __FFLASFFPACK_CONFIGURATION	1053
13.261.1.2 __FFLASFFPACK_HAVE_LAPACK	1053
13.261.2 Function Documentation	1053
13.261.2.1 main()	1053
13.262 regression-check.C File Reference	1053
13.262.1 Function Documentation	1053
13.262.1.1 check1()	1053
13.262.1.2 check2()	1054
13.262.1.3 check3()	1054
13.262.1.4 check4()	1054
13.262.1.5 checkZeroDimCharpoly()	1054
13.262.1.6 checkZeroDimMinPoly()	1054
13.262.1.7 gf2ModularBalanced()	1054
13.262.1.8 main()	1054
13.263 test-charpoly-check.C File Reference	1054
13.263.1 Macro Definition Documentation	1054
13.263.1.1 ENABLE_CHECKER_charpoly	1054
13.263.1.2 TIME_CHECKER_CHARPOLY	1054

13.263.2 Function Documentation	1055
13.263.2.1 printPolynomial()	1055
13.263.2.2 main()	1055
13.264 test-charpoly.C File Reference	1055
13.264.1 Function Documentation	1055
13.264.1.1 launch_test()	1055
13.264.1.2 run_with_field()	1055
13.264.1.3 main()	1056
13.265 test-compressQ.C File Reference	1056
13.265.1 Typedef Documentation	1056
13.265.1.1 Field	1056
13.265.2 Function Documentation	1056
13.265.2.1 printvect()	1056
13.265.2.2 main()	1056
13.266 test-det-check.C File Reference	1057
13.266.1 Macro Definition Documentation	1057
13.266.1.1 ENABLE_CHECKER_Det	1057
13.266.1.2 TIME_CHECKER_Det	1057
13.266.2 Function Documentation	1057
13.266.2.1 main()	1057
13.267 test-det.C File Reference	1057
13.267.1 Function Documentation	1058
13.267.1.1 test_det()	1058
13.267.1.2 main()	1058
13.268 test-echelon.C File Reference	1058
13.268.1 Macro Definition Documentation	1059
13.268.1.1 __FFLASFFPACK_SEQUENTIAL	1059
13.268.1.2 __FFLASFFPACK_GAUSSJORDAN_BASECASE	1059
13.268.1.3 __FFLASFFPACK_PLUQ_THRESHOLD	1059
13.268.2 Function Documentation	1059
13.268.2.1 test_colechelon()	1059
13.268.2.2 test_rowechelon()	1059
13.268.2.3 test_redcoechelon()	1059
13.268.2.4 test_redrowechelon()	1060
13.268.2.5 run_with_field()	1060
13.268.2.6 main()	1060
13.269 test-fadd.C File Reference	1060
13.269.1 Function Documentation	1061
13.269.1.1 test_fadd()	1061
13.269.1.2 test_faddin()	1061
13.269.1.3 test_fsub()	1061
13.269.1.4 test_fsubin()	1061

13.269.1.5 main()	1061
13.270 test-fdot.C File Reference	1061
13.270.1 Macro Definition Documentation	1062
13.270.1.1 ENABLE_ALL_CHECKINGS	1062
13.270.2 Function Documentation	1062
13.270.2.1 check_fdot()	1062
13.270.2.2 run_with_field()	1062
13.270.2.3 run_with_Integer()	1062
13.270.2.4 main()	1063
13.271 test-fgemm-check.C File Reference	1063
13.271.1 Macro Definition Documentation	1063
13.271.1.1 ENABLE_ALL_CHECKINGS	1063
13.271.2 Function Documentation	1063
13.271.2.1 launch_MM_dispatch()	1063
13.271.2.2 run_with_field()	1064
13.271.2.3 main()	1064
13.272 test-fgemm.C File Reference	1064
13.272.1 Macro Definition Documentation	1065
13.272.1.1 ENABLE_CHECKER_fgemm	1065
13.272.2 Function Documentation	1065
13.272.2.1 check_MM()	1065
13.272.2.2 launch_MM()	1065
13.272.2.3 launch_MM_dispatch()	1065
13.272.2.4 run_with_field()	1066
13.272.2.5 main()	1066
13.273 test-fgemv.C File Reference	1066
13.273.1 Function Documentation	1067
13.273.1.1 check_MV()	1067
13.273.1.2 launch_MV()	1067
13.273.1.3 launch_MV_dispatch()	1067
13.273.1.4 run_with_field()	1068
13.273.1.5 main()	1068
13.274 test-fger.C File Reference	1068
13.274.1 Macro Definition Documentation	1068
13.274.1.1 TIME	1068
13.274.2 Function Documentation	1069
13.274.2.1 check_fger()	1069
13.274.2.2 launch_fger()	1069
13.274.2.3 launch_fger_dispatch()	1069
13.274.2.4 run_with_field()	1069
13.274.2.5 main()	1070
13.275 test-fgesv.C File Reference	1070



13.275.1 Function Documentation	1070
13.275.1.1 test_square_fgesv()	1070
13.275.1.2 test_rect_fgesv()	1070
13.275.1.3 run_with_field()	1071
13.275.1.4 main()	1071
13.276 test-finit.C File Reference	1071
13.276.1 Function Documentation	1071
13.276.1.1 test_freduce()	1071
13.276.1.2 run_with_field()	1072
13.276.1.3 main()	1072
13.277 test-fscal.C File Reference	1072
13.277.1 Function Documentation	1072
13.277.1.1 test_fscal() [1/2]	1072
13.277.1.2 test_fscal() [2/2]	1073
13.277.1.3 test_fscalin() [1/2]	1073
13.277.1.4 test_fscalin() [2/2]	1073
13.277.1.5 main()	1073
13.278 test-fsyr2k.C File Reference	1073
13.278.1 Macro Definition Documentation	1074
13.278.1.1 ENABLE_ALL_CHECKINGS	1074
13.278.2 Function Documentation	1074
13.278.2.1 check_fsyr2k()	1074
13.278.2.2 run_with_field()	1074
13.278.2.3 main()	1074
13.279 test-fsyrr.C File Reference	1074
13.279.1 Macro Definition Documentation	1075
13.279.1.1 ENABLE_ALL_CHECKINGS	1075
13.279.2 Function Documentation	1075
13.279.2.1 check_fsyrr()	1075
13.279.2.2 check_fsyrr_diag()	1075
13.279.2.3 check_fsyrr_bkdiag()	1076
13.279.2.4 check_computeS1S2()	1076
13.279.2.5 run_with_field()	1076
13.279.2.6 main()	1076
13.280 test-fsytrf.C File Reference	1076
13.280.1 Function Documentation	1077
13.280.1.1 operator<<()	1077
13.280.1.2 test_RPM_fsytrf()	1077
13.280.1.3 test_generic_fsytrf()	1077
13.280.1.4 run_with_field()	1077
13.280.1.5 main()	1078
13.281 test-ftmrm.C File Reference	1078

13.281.1 Macro Definition Documentation	1078
13.281.1.1 __FFLASFFPACK_SEQUENTIAL	1078
13.281.2 Function Documentation	1078
13.281.2.1 check_ftrmm()	1078
13.281.2.2 run_with_field()	1079
13.281.2.3 main()	1079
13.282 test-ftrmv.C File Reference	1079
13.282.1 Macro Definition Documentation	1079
13.282.1.1 __FFLASFFPACK_SEQUENTIAL	1079
13.282.1.2 ENABLE_ALL_CHECKINGS	1079
13.282.2 Function Documentation	1080
13.282.2.1 check_ftrmv()	1080
13.282.2.2 run_with_field()	1080
13.282.2.3 main()	1080
13.283 test-ftrsm-check.C File Reference	1080
13.283.1 Macro Definition Documentation	1080
13.283.1.1 ENABLE_ALL_CHECKINGS	1080
13.283.2 Function Documentation	1080
13.283.2.1 main()	1080
13.284 test-ftrsm.C File Reference	1081
13.284.1 Macro Definition Documentation	1081
13.284.1.1 __FFLASFFPACK_SEQUENTIAL	1081
13.284.1.2 ENABLE_ALL_CHECKINGS	1081
13.284.2 Function Documentation	1081
13.284.2.1 check_ftrsm()	1081
13.284.2.2 run_with_field()	1082
13.284.2.3 main()	1082
13.285 test-ftrssyr2k.C File Reference	1082
13.285.1 Macro Definition Documentation	1082
13.285.1.1 ENABLE_ALL_CHECKINGS	1082
13.285.2 Function Documentation	1082
13.285.2.1 check_ftrssyr2k()	1082
13.285.2.2 run_with_field()	1083
13.285.2.3 main()	1083
13.286 test-ftrstr.C File Reference	1083
13.286.1 Macro Definition Documentation	1083
13.286.1.1 ENABLE_ALL_CHECKINGS	1083
13.286.2 Function Documentation	1084
13.286.2.1 check_ftrstr()	1084
13.286.2.2 run_with_field()	1084
13.286.2.3 main()	1084
13.287 test-ftrsv.C File Reference	1084

13.287.1 Macro Definition Documentation	1085
13.287.1.1 __FFLASFFPACK_SEQUENTIAL	1085
13.287.1.2 ENABLE_ALL_CHECKINGS	1085
13.287.2 Function Documentation	1085
13.287.2.1 check_ftsv()	1085
13.287.2.2 run_with_field()	1085
13.287.2.3 main()	1085
13.288 test-fttri.C File Reference	1085
13.288.1 Macro Definition Documentation	1086
13.288.1.1 __FFLASFFPACK_SEQUENTIAL	1086
13.288.1.2 ENABLE_ALL_CHECKINGS	1086
13.288.2 Function Documentation	1086
13.288.2.1 check_fttri()	1086
13.288.2.2 run_with_field()	1086
13.288.2.3 main()	1086
13.289 test-interfaces-c.c File Reference	1086
13.289.1 Function Documentation	1087
13.289.1.1 main()	1087
13.290 test-invert-check.C File Reference	1087
13.290.1 Macro Definition Documentation	1087
13.290.1.1 ENABLE_ALL_CHECKINGS	1087
13.290.2 Function Documentation	1087
13.290.2.1 main()	1087
13.291 test-io.C File Reference	1087
13.291.1 Function Documentation	1088
13.291.1.1 run_with_field()	1088
13.291.1.2 main()	1088
13.292 test-lu.C File Reference	1088
13.292.1 Macro Definition Documentation	1089
13.292.1.1 BASECASE_K	1089
13.292.1.2 __FFLASFFPACK_SEQUENTIAL	1089
13.292.1.3 __LUDIVINE_CUTOFF	1089
13.292.2 Function Documentation	1089
13.292.2.1 test_LUdivine()	1089
13.292.2.2 verifPLUQ()	1090
13.292.2.3 test_pluq()	1090
13.292.2.4 launch_test()	1091
13.292.2.5 run_with_field()	1091
13.292.2.6 main()	1091
13.292.3 Variable Documentation	1091
13.292.3.1 tperm	1091
13.292.3.2 tgemm	1091

13.292.3.3 tBC	1092
13.292.3.4 ttrsm	1092
13.292.3.5 trest	1092
13.292.3.6 timtot	1092
13.292.3.7 mvcnt	1092
13.293 test-maxdelayeddim.C File Reference	1092
13.293.1 Macro Definition Documentation	1092
13.293.1.1 MAX_WITH_SIZE_T	1092
13.293.2 Function Documentation	1092
13.293.2.1 test()	1092
13.293.2.2 main()	1092
13.294 test-minpoly.C File Reference	1093
13.294.1 Function Documentation	1093
13.294.1.1 check_minpoly()	1093
13.294.1.2 run_with_field()	1093
13.294.1.3 main()	1093
13.295 test-multifile1.C File Reference	1093
13.296 test-multifile2.C File Reference	1094
13.296.1 Function Documentation	1094
13.296.1.1 main()	1094
13.297 test-nullspace.C File Reference	1094
13.297.1 Function Documentation	1094
13.297.1.1 checkingMessage()	1094
13.297.1.2 readOrRandomMatrixWithRankAndRandomRPM()	1094
13.297.1.3 test_nullspace()	1095
13.297.1.4 run_with_field()	1095
13.297.1.5 main()	1095
13.298 test-permutations.C File Reference	1095
13.298.1 Function Documentation	1096
13.298.1.1 checkMonotonicApplyP()	1096
13.298.1.2 main()	1096
13.298.2 Variable Documentation	1096
13.298.2.1 tperm	1096
13.298.2.2 tgemm	1096
13.298.2.3 tBC	1096
13.298.2.4 ttrsm	1096
13.298.2.5 trest	1096
13.298.2.6 timtot	1096
13.299 test-pluq-check.C File Reference	1096
13.299.1 Macro Definition Documentation	1097
13.299.1.1 ENABLE_ALL_CHECKINGS	1097
13.299.2 Function Documentation	1097

13.299.2.1 main()	1097
13.300 test-quasisep.C File Reference	1097
13.300.1 Function Documentation	1097
13.300.1.1 test_BruhatGenerator()	1097
13.300.1.2 launch_test()	1098
13.300.1.3 testLTQSRPM()	1098
13.300.1.4 run_with_field()	1098
13.300.1.5 main()	1098
13.301 test-rankprofiles.C File Reference	1098
13.301.1 Macro Definition Documentation	1099
13.301.1.1 __FFLASFFPACK_SEQUENTIAL	1099
13.301.2 Function Documentation	1099
13.301.2.1 run_with_field()	1099
13.301.2.2 main()	1099
13.302 test-rpm.C File Reference	1099
13.302.1 Function Documentation	1099
13.302.1.1 checkRPM()	1099
13.302.1.2 checkSymmetricRPM()	1100
13.302.1.3 main()	1100
13.303 test-simd.C File Reference	1100
13.303.1 Macro Definition Documentation	1101
13.303.1.1 _TEST_ONE	1101
13.303.1.2 TEST_ONE_OP	1101
13.303.1.3 TEST_ONE_OP_WZ	1101
13.303.1.4 TEST_IMPL	1102
13.303.2 Function Documentation	1102
13.303.2.1 check_eq() [1/2]	1102
13.303.2.2 check_eq() [2/2]	1102
13.303.2.3 cmp()	1102
13.303.2.4 eval_func_on_array() [1/3]	1102
13.303.2.5 eval_func_on_array() [2/3]	1102
13.303.2.6 eval_func_on_array() [3/3]	1102
13.303.2.7 operator<<()	1102
13.303.2.8 test_impl_base() [1/2]	1103
13.303.2.9 test_impl_base() [2/2]	1103
13.303.2.10 test_impl()	1103
13.303.2.11 main()	1103
13.304 test-solve.C File Reference	1103
13.304.1 Function Documentation	1103
13.304.1.1 check_solve()	1103
13.304.1.2 run_with_field()	1103
13.304.1.3 main()	1104

13.305 test-storage-transpose.C File Reference	1104
13.305.1 Function Documentation	1104
13.305.1.1 main()	1104
13.306 101-fgemm.C File Reference	1104
13.306.1 Function Documentation	1105
13.306.1.1 main()	1105
13.307 2x2-fgemm.C File Reference	1105
13.307.1 Function Documentation	1105
13.307.1.1 main()	1105
13.308 2x2-ftrsv.C File Reference	1105
13.308.1 Function Documentation	1105
13.308.1.1 main()	1105
13.309 2x2-pluq.C File Reference	1106
13.309.1 Function Documentation	1106
13.309.1.1 main()	1106
13.310 fflas-101_1.C File Reference	1106
13.310.1 Function Documentation	1106
13.310.1.1 main()	1106
13.311 fflas-101_3.C File Reference	1106
13.311.1 Function Documentation	1107
13.311.1.1 main()	1107
13.312 fflas_101.C File Reference	1107
13.312.1 Function Documentation	1107
13.312.1.1 main()	1107
13.313 fflas_101_lvl1.C File Reference	1107
13.313.1 Function Documentation	1107
13.313.1.1 main()	1107
13.314 ffpack-fgesv.C File Reference	1107
13.314.1 Function Documentation	1108
13.314.1.1 main()	1108
13.315 ffpack-solve.C File Reference	1108
13.315.1 Function Documentation	1108
13.315.1.1 main()	1108

# Chapter 1

## FFLAS-FFPACK Documentation.

### 1.1 Introduction

FFLAS-FFPACK is a LGPL-2.1+ source code library for basic linear algebra operations over a finite field. It is inspired by BLAS interface (Basic Linear Algebra Subprograms) and the LAPACK library for numerical linear algebra, and shares part of their design. Yet it differs in many aspects due to the specifics of computing over a finite field:

- it is generic with respect to the finite field, so as to accomodate a large variety of field sizes and implementations;
- it is a pure source code library, to be included and compiled in the user's software. Its build system is only used for tests and benchmarks.

### 1.2 Goals

### 1.3 Design

### 1.4 Using FFLAS-FFPACK.

- [Copying and Licence](#).
- [Tutorial](#). This is a brief introduction to FFLAS-FFPACK capabilities.
- [Configuring and Installing FFLAS-FFPACK](#). Explains how to configure/install from sources or from the latest svn version.
- [Architecture of the library](#).. Describes how FFLAS-FFPACK is organized
- [Documentation for Users](#). If everything around is blue, then you are reading the lighter, user-oriented, documentation.
- [Documentation for Developers](#). If everything around is green, then you can get to everything (not necessarily yet) documented.

## 1.5 Contributing to fflas-ffpack, getting assistance.

Version

2.5.0

## 1.6 Copying and Licence

The FFLAS-FFPACK library is licensed under the terms of the GNU LGPL v2.1 or later.

See <https://www.gnu.org/licenses/lgpl-2.1.html>

## 1.7 Tutorial

no doc.

## 1.8 Configuring and Installing FFLAS-FFPACK

FFLAS-FFPACK is a header-only package.

Howver configuration process can be tweaked a lot. Configure looks for BLAS routines and [Givaro](#) library which are both mandatory dependencies. See the output of `./configure -help` for information about the LAPACK/↔ BLAS discovering strategies.

## 1.9 Architecture of the library.

no doc.



## Chapter 2

# Bug List

Global **DOUBLE\_TO\_FLOAT\_CROSSOVER**

to be benchmarked.

Global **FFLAS::details::pack\_lhs** (int64\_t \*XX, const int64\_t \*X, size\_t ldx, size\_t rows, size\_t cols)

this is fassign

this is fassign

Global **FFLAS::details::pack\_rhs** (int64\_t \*XX, const int64\_t \*X, size\_t ldx, size\_t rows, size\_t cols)

this is fassign

this is fassign

Global **FFLAS::fconvert** (const Field &F, const size\_t n, OtherElement\_ptr X, const size\_t incX, typename **Field::ConstElement\_ptr** Y, const size\_t incY)

use cblas\_(d)scal when possible

Global **FFLAS::fconvert** (const FFLAS\_FIELD< FFLAS\_ELT > &F, const size\_t n, FFLAS\_ELT \*X, const size\_t incX, const FFLAS\_ELT \*Y, const size\_t incY)

use cblas\_(d)scal when possible

Global **FFLAS::finit** (const Field &F, const size\_t n, const OtherElement\_ptr Y, const size\_t incY, typename **Field::Element\_ptr** X, const size\_t incX)

use cblas\_(d)scal when possible

Global **FFLAS::finit** (const FFLAS\_FIELD< FFLAS\_ELT > &F, const size\_t n, const FFLAS\_ELT \*Y, const size\_t incY, FFLAS\_ELT \*X, const size\_t incX)

use cblas\_(d)scal when possible

Global **FFLAS::fneg** (const Field &F, const size\_t n, typename **Field::ConstElement\_ptr** Y, const size\_t incY, typename **Field::Element\_ptr** X, const size\_t incX)

use cblas\_(d)scal when possible

Global **FFLAS::fneg** (const FFLAS\_FIELD< FFLAS\_ELT > &F, const size\_t n, const FFLAS\_ELT \*Y, const size\_t incY, FFLAS\_ELT \*X, const size\_t incX)

use cblas\_(d)scal when possible

Global **FFLAS::fnegin** (const Field &F, const size\_t n, typename **Field::Element\_ptr** X, const size\_t incX)

use cblas\_(d)scal when possible

Global **FFLAS::fnegin** (const FFLAS\_FIELD< FFLAS\_ELT > &F, const size\_t n, FFLAS\_ELT \*X, const size\_t incX)

use cblas\_(d)scal when possible

Global **FFLAS::freduce** (const Field &F, const size\_t n, typename Field::Element\_ptr X, const size\_t incX)

use cblas\_(d)scal when possible

Global **FFLAS::freduce** (const Field &F, const size\_t n, typename Field::ConstElement\_ptr Y, const size\_t incY, typename Field::Element\_ptr X, const size\_t incX)

use cblas\_(d)scal when possible

Global **FFLAS::freduce** (const FFLAS\_FIELD< FFLAS\_ELT > &F, const size\_t n, FFLAS\_ELT \*X, const size\_t incX)

use cblas\_(d)scal when possible

Global **FFLAS::freduce** (const FFLAS\_FIELD< FFLAS\_ELT > &F, const size\_t n, const FFLAS\_ELT \*Y, const size\_t incY, FFLAS\_ELT \*X, const size\_t incX)

use cblas\_(d)scal when possible

Global **FFLAS::fscal** (const Field &F, const size\_t n, const typename Field::Element alpha, typename Field::ConstElement\_ptr X, const size\_t incX, typename Field::Element\_ptr Y, const size\_t incY)

use cblas\_(d)scal when possible

Global **FFLAS::fscal** (const FFLAS\_FIELD< FFLAS\_ELT > &F, const size\_t n, const FFLAS\_ELT alpha, const FFLAS\_ELT \*X, const size\_t incX, FFLAS\_ELT \*Y, const size\_t incY)

use cblas\_(d)scal when possible

Global **FFLAS::fscaln** (const Field &F, const size\_t n, const typename Field::Element alpha, typename Field::Element\_ptr X, const size\_t incX)

use cblas\_(d)scal when possible

Global **FFLAS::fscaln** (const FFLAS\_FIELD< FFLAS\_ELT > &F, const size\_t n, const FFLAS\_ELT alpha, FFLAS\_ELT \*X, const size\_t incX)

use cblas\_(d)scal when possible

Global **FFLAS::fsquare** (const Field &F, const FFLAS\_TRANSPOSE ta, const size\_t n, const typename Field::Element alpha, typename Field::ConstElement\_ptr A, const size\_t lda, const typename Field::Element beta, typename Field::Element\_ptr C, const size\_t ldc)

why double ?

Global **FFLAS::fswap** (const Field &F, const size\_t N, typename Field::Element\_ptr X, const size\_t incX, typename Field::Element\_ptr Y, const size\_t incY)

use cblas\_dswap when double

Global **FFLAS::fswap** (const FFLAS\_FIELD< FFLAS\_ELT > &F, const size\_t N, FFLAS\_ELT \*X, const size\_t incX, FFLAS\_ELT \*Y, const size\_t incY)

use cblas\_dswap when double

Global **FFLAS::ftrsm** (const Field &F, const FFLAS\_SIDE Side, const FFLAS\_UPLO Uplo, const FFLAS\_TRANSPOSE TransA, const FFLAS\_DIAG Diag, const size\_t M, const size\_t N, const typename Field::Element alpha, typename Field::ConstElement\_ptr A, const size\_t lda, typename Field::Element\_ptr B, const size\_t ldb)

$\alpha$  must be non zero.

---

Global **FFLAS::ftrsm** (const FFLAS\_FIELD< FFLAS\_ELT > &F, const FFLAS\_SIDE Side, const FFLAS\_UPLO Uplo, const FFLAS\_TRANSPOSE TransA, const FFLAS\_DIAG Diag, const size\_t M, const size\_t N, const FFLAS\_ELT alpha, const FFLAS\_ELT \*A, const size\_t lda, FFLAS\_ELT \*B, const size\_t ldb)

$\alpha$  must be non zero.

Global **FFPACK::buildMatrix** (const Field &F, typename **Field::ConstElement\_ptr** E, typename **Field::ConstElement\_ptr** C, const size\_t lda, const size\_t \*B, const size\_t \*T, const size\_t me, const size\_t mc, const size\_t lambda, const size\_t mu)

is this :

Global **FFPACK::invert2** (const Field &F, const size\_t M, typename **Field::Element\_ptr** A, const size\_t lda, typename **Field::Element\_ptr** X, const size\_t ldx, int &>nullity)

not tested.

Global **launch\_fger\_dispatch** (const Field &F, const size\_t nn, const typename **Field::Element** alpha, const size\_t iters, RandIter &G)

test for incx equal

test for transpo

Global **launch\_MM\_dispatch** (const Field &F, const int mm, const int nn, const int kk, const typename **Field::Element** alpha, const typename **Field::Element** beta, const size\_t iters, RandIter &G)

test for ldX equal

test for transpo

Global **launch\_MM\_dispatch** (const Field &F, const int mm, const int nn, const int kk, const typename **Field::Element** alpha, const typename **Field::Element** beta, const size\_t iters, const int nbw, const bool par, RandIter &G)

test for ldX equal

test for transpo

Global **printvect** (std::ostream &o, vector< T > &vect)

does not belong here



## Chapter 3

# Bibliography

Global **FFLAS::Protected::TRSMBound** (const Givaro::ModularBalanced< Element > &F)

- Dumas Giorgi Pernet 06, arXiv:cs/0601133

Global **FFPACK::LeadingSubmatrixRankProfiles** (const size\_t M, const size\_t N, const size\_t R, const size\_t LSm, const size\_t LSn, const size\_t \*P, const size\_t \*Q, size\_t \*RRP, size\_t \*CRP)

- Dumas J-G., Pernet C., and Sultan Z. *Simultaneous computation of the row and column rank profiles*, ISSAC'13.

Global **FFPACK::LUdivine** (const Field &F, const **FFLAS::FFLAS\_DIAG** Diag, const **FFLAS::FFLAS\_TRANSPOSE** trans, const size\_t M, const size\_t N, typename **Field::Element\_ptr** A, const size\_t lda, size\_t \*P, size\_t \*Qt, const FFPACK\_LU\_TAG LuTag=FfpackSlabRecursive, const size\_t cutoff=\_\_FFLASFFPACK\_LUDIVINE\_THRESHOLD)

- Jeannerod C-P, Pernet, C. and Storjohann, A. *Rank-profile revealing Gaussian elimination and the CUP matrix decomposition*, J. of Symbolic Comp., 2013
- Pernet C, Brassel M *LUdivine, une divine factorisation LU*, 2002

Global **FFPACK::PLUQ** (const Field &F, const **FFLAS::FFLAS\_DIAG** Diag, const size\_t M, const size\_t N, typename **Field::Element\_ptr** A, const size\_t lda, size\_t \*P, size\_t \*Q)

- Dumas J-G., Pernet C., and Sultan Z. *Simultaneous computation of the row and column rank profiles*, ISSAC'13, 2013

Global **FFPACK::productBruhatxTS** (const Field &Fi, size\_t N, size\_t s, size\_t r, size\_t t, const size\_t \*P, const size\_t \*Q, typename **Field::ConstElement\_ptr** Xu, size\_t ldu, size\_t NbBlocksU, const size\_t \*Ku, const size\_t \*Tu, const size\_t \*MU, typename **Field::ConstElement\_ptr** XI, size\_t ldl, size\_t NbBlocksL, const size\_t \*KI, const size\_t \*TI, const size\_t \*ML, typename **Field::Element\_ptr** B, size\_t ldb, const typename **Field::Element** beta, typename **Field::Element\_ptr** D, size\_t ldd)

Pernet C. and Storjohann A. *Time and space efficient generators for quasiseparable matrices*, JSC (85), 2018, doi:10.1016/j.jsc.2017.07.010

Global `FFPACK::Protected::GaussJordan` (const Field &F, const size\_t M, const size\_t N, typename Field::Element\_ptr A, const size\_t lda, const size\_t colbeg, const size\_t rowbeg, const size\_t colsize, size\_t \*P, size\_t \*Q, const FFPACK::FFPACK\_LU\_TAG LuTag)

- Algorithm 2.8 of A. Storjohann Thesis 2000,
- Algorithm 11 of Jeannerod C-P., Pernet, C. and Storjohann, A. *Rank-profile revealing Gaussian elimination and the CUP matrix decomposition*, J. of Symbolic Comp., 2013

Class `ftsrmlLeftUpperNoTransNonUnit` < Element >

- Dumas, Giorgi, Pernet 06, arXiv:cs/0601133.

## Chapter 4

# Todo List

File [debug.h](#)

we should put vector printing elsewhere.

Global [FFLAS::fadd](#) (const Field &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t inca, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) B, const size\_t incb, typename [Field::Element\\_ptr](#) C, const size\_t incc)

optimise here

Global [FFLAS::fassign](#) (const Field &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) Y, const size\_t incY, typename [Field::Element\\_ptr](#) X, const size\_t incX)

variant for triangular matrix

Global [FFLAS::fassign](#) (const FFLAS\_FIELD< FFLAS\_ELT > &F, const size\_t N, const FFLAS\_ELT \*Y, const size\_t incY, FFLAS\_ELT \*X, const size\_t incX)

variant for triangular matrix

Global [FFLAS::fconvert](#) (const Field &F, const size\_t m, const size\_t n, OtherElement\_ptr A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb)

check if n == lda

Global [FFLAS::fneg](#) (const Field &F, const size\_t m, const size\_t n, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) A, const size\_t lda)

check if n == lda

Global [FFLAS::fnegin](#) (const Field &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)

check if n == lda

Global [FFLAS::fscal](#) (const Field &F, const size\_t n, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, typename [Field::Element\\_ptr](#) Y, const size\_t incY)

check if comparison with +/-1,0 is necessary.

Global [FFLAS::fscal](#) (const FFLAS\_FIELD< FFLAS\_ELT > &F, const size\_t n, const FFLAS\_ELT alpha, const FFLAS\_ELT \*X, const size\_t incX, FFLAS\_ELT \*Y, const size\_t incY)

check if comparison with +/-1,0 is necessary.

Global [FFLAS::fscalin](#) (const Field &F, const size\_t n, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) X, const size\_t incX)

check if comparison with +/-1,0 is necessary.

Global **FFLAS::fscaln** (const FFLAS\_FIELD< FFLAS\_ELT > &F, const size\_t n, const FFLAS\_ELT alpha, FFLAS\_ELT \*X, const size\_t incX)

check if comparison with +/-1,0 is necessary.

Global **FFLAS::Protected::igemm** (const enum FFLAS\_TRANSPOSE TransA, const enum FFLAS\_TRANSPOSE TransB, size\_t rows, size\_t cols, size\_t depth, const int64\_t alpha, const int64\_t \*A, size\_t lda, const int64\_t \*B, size\_t ldb, const int64\_t beta, int64\_t \*C, size\_t ldc)

use primitive (no **Field()**) and specialise for int64.

Global **FFLAS::Protected::MatF2MatFI\_Triangular** (const Field &F, Givaro::FloatDomain::Element\_ptr S, const size\_t lds, typename **Field::ConstElement\_ptr** const E, const size\_t lde, const size\_t m, const size\_t n)

do finit(...,FFLAS\_TRANS,FFLAS\_DIAG)

do fconvert(...,FFLAS\_TRANS,FFLAS\_DIAG)

Global **FFPACK::getTriangular** (const Field &F, const **FFLAS::FFLAS\_UPLO** Uplo, const **FFLAS::FFLAS\_DIAG** diag, const size\_t M, const size\_t N, const size\_t R, typename **Field::ConstElement\_ptr** A, const size\_t lda, typename **Field::Element\_ptr** T, const size\_t ldt, const bool OnlyNonZeroVectors=false)

just one triangular fzero+fassign ?

Global **FFPACK::getTriangular** (const Field &F, const **FFLAS::FFLAS\_UPLO** Uplo, const **FFLAS::FFLAS\_DIAG** diag, const size\_t M, const size\_t N, const size\_t R, typename **Field::Element\_ptr** A, const size\_t lda)

just one triangular fzero+fassign ?

Global **FFPACK::Invert2** (const Field &F, const size\_t M, typename **Field::Element\_ptr** A, const size\_t lda, typename **Field::Element\_ptr** X, const size\_t ldx, int &>nullity)

this init is not all necessary (done after ftrtri)

Global **FFPACK::LUdivine** (const Field &F, const **FFLAS::FFLAS\_DIAG** Diag, const **FFLAS::FFLAS\_TRANSPOSE** trans, const size\_t M, const size\_t N, typename **Field::Element\_ptr** A, const size\_t lda, size\_t \*P, size\_t \*Q, const FFPACK::FFPACK\_LU\_TAG LuTag, const size\_t cutoff)

std::swap ?

Global **FFPACK::Protected::RandomKrylovPrecond** (const PolRing &PR, std::list< typename PolRing::Element > &completedFactors, const size\_t N, typename PolRing::Domain\_t::Element\_ptr A, const size\_t lda, size\_t &Nb, typename PolRing::Domain\_t::Element\_ptr &B, size\_t &ldb, typename PolRing::Domain\_t::RandIter &g, const size\_t degree=\_\_FFLASFFPACK\_ARITHPROG\_THRESHOLD)

swap to save space ??

don't assing K2 c\*noc x N but only mas (c,noc) x N and store each one after the other

## Module **field**

biblio

Global **launch\_fger\_dispatch** (const Field &F, const size\_t nn, const typename **Field::Element** alpha, const size\_t iters, RandIter &G)

does nbw actually do nbw recursive calls and then call blas (check ?) ?

Global **launch\_MM\_dispatch** (const Field &F, const int mm, const int nn, const int kk, const typename **Field::Element** alpha, const typename **Field::Element** beta, const size\_t iters, RandIter &G)

does nbw actually do nbw recursive calls and then call blas (check ?) ?



Global `launch_MM_dispatch` (const Field &F, const int mm, const int nn, const int kk, const typename `Field::Element` alpha, const typename `Field::Element` beta, const size\_t iters, const int nbw, const bool par, RandIter &G)

does nbw actually do nbw recursive calls and then call blas (check ?) ?

Module `MMalgos`

biblio

Module `simd`

biblio

Global `test_colechelon` (Field &F, size\_t m, size\_t n, size\_t r, size\_t iters, `FFPACK::FFPACK_LU_TAG` LuTag, RandIter &G, bool par)

check Ida

Global `test_det` (Field &F, size\_t n, int iter, RandIter &G)

test with stride

Global `test_redcolechelon` (Field &F, size\_t m, size\_t n, size\_t r, size\_t iters, `FFPACK::FFPACK_LU_TAG` LuTag, RandIter &G, bool par)

check Ida

Global `test_redrowechelon` (Field &F, size\_t m, size\_t n, size\_t r, size\_t iters, `FFPACK::FFPACK_LU_TAG` LuTag, RandIter &G, bool par)

check Ida

Global `test_rowechelon` (Field &F, size\_t m, size\_t n, size\_t r, size\_t iters, `FFPACK::FFPACK_LU_TAG` LuTag, RandIter &G, bool par)

check Ida



# Chapter 5

## Topic Index

### 5.1 Topics

Here is a list of all topics with brief descriptions:

CHECKER . . . . .	41
FFLAS-FFPACK . . . . .	41
FFLAS . . . . .	41
Interfaces . . . . .	42
Matrix Multiplication Algorithms . . . . .	42
SIMD wrapper . . . . .	42
FFPACK . . . . .	42
FFLAS-FFPACK fields . . . . .	43
RNS . . . . .	43



## Chapter 6

# Namespace Index

### 6.1 Namespace List

Here is a list of all namespaces with brief descriptions:

FFLAS	45
FFLAS::_ftranspose_impl	190
FFLAS::BLAS3	191
FFLAS::csr_hyb_details	197
FFLAS::CuttingStrategy	197
FFLAS::details	198
FFLAS::details_spmv	206
FFLAS::ElementCategories	207
FFLAS::FieldCategories	
Traits and categories will need to be placed in a proper file later	207
FFLAS::MMHelperAlgo	207
FFLAS::ModeCategories	
Specifies the mode of action for an algorithm w.r.t	208
FFLAS::ParSeqHelper	
ParSeqHelper for both fgemm and ftrsm	208
FFLAS::Protected	209
FFLAS::sell_details	223
FFLAS::sparse_details	223
FFLAS::sparse_details_impl	237
FFLAS::StrategyParameter	278
FFLAS::StructureHelper	
StructureHelper for ftrsm	278
FFLAS::vectorised	278
FFLAS::vectorised::unswitch	284
FFPACK	
Finite Field <b>PACK</b> Set of elimination based routines for dense linear algebra	287
FFPACK::Protected	391
Givaro	398
MKL_CONFIG	398
RecInt	398



# Chapter 7

## Hierarchical Index

### 7.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

AlgoChooser< ModeT, ParSeq > . . . . .	401
AlgoChooser< ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq > . . . . .	401
ALL< v > . . . . .	401
ALL< false, v... > . . . . .	402
ALL< true, v... > . . . . .	402
ALL<> . . . . .	402
ArbitraryPrecIntTag . . . . .	402
AreEqual< X, Y > . . . . .	403
AreEqual< X, X > . . . . .	403
Argument . . . . .	403
associatedDelayedField< Field > . . . . .	404
associatedDelayedField< const FFPACK::RNSIntegerMod< RNS > > . . . . .	404
associatedDelayedField< const Givaro::Modular< T, X > > . . . . .	405
associatedDelayedField< const Givaro::ModularBalanced< T > > . . . . .	405
associatedDelayedField< const Givaro::ZRing< T > > . . . . .	406
Auto . . . . .	406
Bench< Elt > . . . . .	406
Bini . . . . .	409
Block . . . . .	409
BlockTransposeSIMD< Field, Simd, > . . . . .	409
callLUdivine_small< Element > . . . . .	411
callLUdivine_small< double > . . . . .	411
callLUdivine_small< float > . . . . .	412
CharpolyFailed . . . . .	412
Checker_Empty< Field > . . . . .	412
CheckerImplem_charpoly< Field, Polynomial > . . . . .	413
CheckerImplem_charpoly< Givaro::ZRing< Givaro::Integer >, Polynomial > . . . . .	414
CheckerImplem_Det< Field > . . . . .	415
CheckerImplem_fgemm< Field > . . . . .	417
CheckerImplem_ftrsm< Field > . . . . .	418
CheckerImplem_invert< Field > . . . . .	419
CheckerImplem_PLUQ< Field > . . . . .	420
Classic . . . . .	421
Column . . . . .	421
CompactElement< Element > . . . . .	421

CompactElement< double > . . . . .	421
CompactElement< float > . . . . .	422
CompactElement< int16_t > . . . . .	422
CompactElement< int32_t > . . . . .	422
CompactElement< int64_t > . . . . .	422
compatible_data_type< Field > . . . . .	423
compatible_data_type< Givaro::ZRing< double > > . . . . .	423
compatible_data_type< Givaro::ZRing< float > > . . . . .	423
Compose< H1, H2 > . . . . .	424
Simd128_impl< true, true, false, 2 >::Converter . . . . .	425
Simd128_impl< true, true, false, 4 >::Converter . . . . .	425
Simd128_impl< true, true, false, 8 >::Converter . . . . .	425
Simd128_impl< true, true, true, 2 >::Converter . . . . .	426
Simd128_impl< true, true, true, 4 >::Converter . . . . .	426
Simd128_impl< true, true, true, 8 >::Converter . . . . .	426
Simd256_impl< true, false, true, 8 >::Converter . . . . .	427
Simd256_impl< true, true, false, 2 >::Converter . . . . .	427
Simd256_impl< true, true, false, 4 >::Converter . . . . .	427
Simd256_impl< true, true, false, 8 >::Converter . . . . .	428
Simd256_impl< true, true, true, 2 >::Converter . . . . .	428
Simd256_impl< true, true, true, 4 >::Converter . . . . .	428
Simd256_impl< true, true, true, 8 >::Converter . . . . .	429
Simd512_impl< true, true, false, 8 >::Converter . . . . .	429
Simd512_impl< true, true, true, 8 >::Converter . . . . .	429
ConvertTo< T > . . . . .	430
Coo< ValT, IdxT > . . . . .	430
Coo< Field > . . . . .	432
Coo< ValT, IdxT > . . . . .	433
CooMat< Field > . . . . .	435
CooMat< FFPACK::RNSInteger > . . . . .	435
count_nonconst_lvalue_reference< T > . . . . .	436
count_nonconst_lvalue_reference< const T &, O... > . . . . .	436
count_nonconst_lvalue_reference< T &, O... > . . . . .	436
count_nonconst_lvalue_reference< T, O... > . . . . .	436
count_nonconst_lvalue_reference<> . . . . .	437
CsrMat< Field > . . . . .	437
CsrMat< FFPACK::RNSInteger > . . . . .	437
DefaultBoundedTag . . . . .	438
DefaultTag . . . . .	438
DelayedTag . . . . .	438
DivideAndConquer . . . . .	438
ElementTraits< Element > . . . . .	438
ElementTraits< double > . . . . .	439
ElementTraits< FFPACK::rns_double_elt > . . . . .	439
ElementTraits< float > . . . . .	439
ElementTraits< Givaro::Integer > . . . . .	440
ElementTraits< int16_t > . . . . .	440
ElementTraits< int32_t > . . . . .	440
ElementTraits< int64_t > . . . . .	440
ElementTraits< int8_t > . . . . .	441
ElementTraits< Reclnt::rint< K > > . . . . .	441
ElementTraits< Reclnt::rmint< K, MG > > . . . . .	441
ElementTraits< Reclnt::ruint< K > > . . . . .	442
ElementTraits< uint16_t > . . . . .	442
ElementTraits< uint32_t > . . . . .	442
ElementTraits< uint64_t > . . . . .	442
ElementTraits< uint8_t > . . . . .	443
EllMat< Field > . . . . .	443



EllMat< FFPACK::RNSInteger > . . . . .	443
Failure . . . . .	444
FailureCharpolyCheck . . . . .	445
FailureDetCheck . . . . .	446
FailureFgemmCheck . . . . .	446
FailureInvertCheck . . . . .	446
FailurePLUQCheck . . . . .	446
FailureTrsmCheck . . . . .	446
false_type	
isSparseMatrix< Field, M > . . . . .	481
isSparseMatrixMKLFormat< F, M > . . . . .	485
isSparseMatrixSimdFormat< F, M > . . . . .	485
isZOSparseMatrix< F, M > . . . . .	486
support_fast_mod< T > . . . . .	759
support_simd< T > . . . . .	760
support_simd_add< T > . . . . .	761
support_simd_mod< T > . . . . .	761
FieldSimd< _Field > . . . . .	446
FieldTraits< Field > . . . . .	452
FieldTraits< FFPACK::RNSInteger< T > > . . . . .	453
FieldTraits< FFPACK::RNSIntegerMod< T > > . . . . .	453
FieldTraits< Givaro::Modular< Element > > . . . . .	454
FieldTraits< Givaro::ModularBalanced< Element > > . . . . .	454
FieldTraits< Givaro::ZRing< double > > . . . . .	455
FieldTraits< Givaro::ZRing< float > > . . . . .	455
FieldTraits< Givaro::ZRing< Givaro::Integer > > . . . . .	456
FieldTraits< Givaro::ZRing< int16_t > > . . . . .	456
FieldTraits< Givaro::ZRing< int32_t > > . . . . .	457
FieldTraits< Givaro::ZRing< int64_t > > . . . . .	457
FieldTraits< Givaro::ZRing< Reclnt::ruint< K > > > . . . . .	458
FieldTraits< Givaro::ZRing< uint16_t > > . . . . .	458
FieldTraits< Givaro::ZRing< uint32_t > > . . . . .	459
FieldTraits< Givaro::ZRing< uint64_t > > . . . . .	459
Fixed . . . . .	460
FixedPreclntTag . . . . .	460
ScalFunctionsBase< Element, typename enable_if< is_floating_point< Element >::value >::type >← ::FloatingPointTestDistribution . . . . .	460
ForStrategy1D< blocksize_t, Cut, Param > . . . . .	461
ForStrategy2D< blocksize_t, Cut, Param > . . . . .	463
ftmmLeftLowerNoTransNonUnit< Element > . . . . .	467
ftmmLeftLowerNoTransUnit< Element > . . . . .	467
ftmmLeftLowerTransNonUnit< Element > . . . . .	467
ftmmLeftLowerTransUnit< Element > . . . . .	467
ftmmLeftUpperNoTransNonUnit< Element > . . . . .	467
ftmmLeftUpperNoTransUnit< Element > . . . . .	467
ftmmLeftUpperTransNonUnit< Element > . . . . .	467
ftmmLeftUpperTransUnit< Element > . . . . .	468
ftmmRightLowerNoTransNonUnit< Element > . . . . .	468
ftmmRightLowerNoTransUnit< Element > . . . . .	468
ftmmRightLowerTransNonUnit< Element > . . . . .	468
ftmmRightLowerTransUnit< Element > . . . . .	468
ftmmRightUpperNoTransNonUnit< Element > . . . . .	468
ftmmRightUpperNoTransUnit< Element > . . . . .	468
ftmmRightUpperTransNonUnit< Element > . . . . .	468
ftmmRightUpperTransUnit< Element > . . . . .	469
ftsmLeftLowerNoTransNonUnit< Element > . . . . .	469
ftsmLeftLowerNoTransUnit< Element > . . . . .	469
ftsmLeftLowerTransNonUnit< Element > . . . . .	469

ftsmLeftLowerTransUnit< Element > . . . . .	469
ftsmLeftUpperNoTransNonUnit< Element > . . . . .	469
ftsmLeftUpperNoTransUnit< Element > . . . . .	470
ftsmLeftUpperTransNonUnit< Element > . . . . .	470
ftsmLeftUpperTransUnit< Element > . . . . .	470
ftsmRightLowerNoTransNonUnit< Element > . . . . .	470
ftsmRightLowerNoTransUnit< Element > . . . . .	470
ftsmRightLowerTransNonUnit< Element > . . . . .	470
ftsmRightLowerTransUnit< Element > . . . . .	470
ftsmRightUpperNoTransNonUnit< Element > . . . . .	470
ftsmRightUpperNoTransUnit< Element > . . . . .	471
ftsmRightUpperTransNonUnit< Element > . . . . .	471
ftsmRightUpperTransUnit< Element > . . . . .	471
GenericTag . . . . .	471
GenericTag . . . . .	471
Grain . . . . .	471
has_minus_eq_impl< C > . . . . .	471
has_minus_impl< C > . . . . .	472
has_mul_eq_impl< C > . . . . .	472
has_mul_impl< C > . . . . .	472
has_operation< T > . . . . .	473
has_plus_eq_impl< C > . . . . .	473
has_plus_impl< C > . . . . .	473
HelperFlag . . . . .	474
HelperMod< Field, ElementTraits > . . . . .	474
HelperMod< Field, ElementCategories::MachineIntTag > . . . . .	475
HelperMod< Field, FFLAS::ElementCategories::ArbitraryPrecIntTag > . . . . .	476
HelperMod< Field, FFLAS::ElementCategories::FixedPrecIntTag > . . . . .	476
HelperMod< Field, FFLAS::ElementCategories::MachineFloatTag > . . . . .	477
Hybrid . . . . .	478
Info . . . . .	478
Info . . . . .	479
is_all_same< Args > . . . . .	480
is_all_same< T, Args... > . . . . .	480
is_all_same<> . . . . .	480
is_simd< T > . . . . .	481
Iterative . . . . .	487
LazyTag . . . . .	487
limits< T > . . . . .	488
limits< char > . . . . .	488
limits< double > . . . . .	488
limits< float > . . . . .	489
limits< Givaro::Integer > . . . . .	490
limits< int > . . . . .	490
limits< long > . . . . .	491
limits< long long > . . . . .	491
limits< Reclnt::rint< K > > . . . . .	492
limits< Reclnt::ruint< K > > . . . . .	493
limits< short int > . . . . .	493
limits< signed char > . . . . .	494
limits< unsigned char > . . . . .	495
limits< unsigned int > . . . . .	495
limits< unsigned long > . . . . .	496
limits< unsigned long long > . . . . .	496
limits< unsigned short int > . . . . .	497
MachineFloatTag . . . . .	498
MachineIntTag . . . . .	498
MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait > . . . . .	498

MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait > . . . . .	503
MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait > . . . . .	505
MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< Dest >, ParSeqTrait > . . . . .	507
MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeqTrait > . . . . .	509
MMHelper< Field, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait > . . . . .	511
ModeTraits< Field > . . . . .	512
ModeTraits< Givaro::Modular< Element, Compute > > . . . . .	513
ModeTraits< Givaro::Modular< Givaro::Integer, Compute > > . . . . .	513
ModeTraits< Givaro::Modular< int16_t, Compute > > . . . . .	513
ModeTraits< Givaro::Modular< int32_t, Compute > > . . . . .	514
ModeTraits< Givaro::Modular< int64_t, uint64_t > > . . . . .	514
ModeTraits< Givaro::Modular< int8_t, Compute > > . . . . .	514
ModeTraits< Givaro::Modular< Reclnt::ruint< K >, Compute > > . . . . .	515
ModeTraits< Givaro::Modular< uint16_t, Compute > > . . . . .	515
ModeTraits< Givaro::Modular< uint32_t, Compute > > . . . . .	515
ModeTraits< Givaro::Modular< uint8_t, Compute > > . . . . .	516
ModeTraits< Givaro::ModularBalanced< Element > > . . . . .	516
ModeTraits< Givaro::ModularBalanced< Givaro::Integer > > . . . . .	516
ModeTraits< Givaro::ModularBalanced< int16_t > > . . . . .	517
ModeTraits< Givaro::ModularBalanced< int32_t > > . . . . .	517
ModeTraits< Givaro::ModularBalanced< int8_t > > . . . . .	517
ModeTraits< Givaro::Montgomery< T > > . . . . .	517
ModeTraits< Givaro::ZRing< double > > . . . . .	518
ModeTraits< Givaro::ZRing< float > > . . . . .	518
ModeTraits< Givaro::ZRing< Givaro::Integer > > . . . . .	518
ModularBalanced< T > . . . . .	519
ModularTag . . . . .	519
Montgomery< T > . . . . .	519
need_field_characteristic< Field > . . . . .	519
need_field_characteristic< Givaro::Modular< Field > > . . . . .	519
need_field_characteristic< Givaro::ModularBalanced< Field > > . . . . .	520
NoSimd< T > . . . . .	520
Parallel< C, P > . . . . .	522
readMyMachineType< Field, T > . . . . .	525
readMyMachineType< Field, mpz_t > . . . . .	525
Recursive . . . . .	526
Recursive . . . . .	526
rint< K > . . . . .	526
rns_double . . . . .	526
rns_double_elt . . . . .	531
rns_double_elt_cstptr . . . . .	533
rns_double_elt_ptr . . . . .	535
rns_double_extended . . . . .	538
RNSElementTag . . . . .	542
RNSInteger< RNS > . . . . .	542
RNSInteger< FFPACK::rns_double > . . . . .	542
RNSIntegerMod< RNS > . . . . .	545
RNSIntegerMod< FFPACK::rns_double > . . . . .	545
rnsRandIter< RNS > . . . . .	552
RNSInteger< RNS >::RandIter . . . . .	523
RNSIntegerMod< RNS >::RandIter . . . . .	524
Row . . . . .	553
ruint< K > . . . . .	553
ScalFunctionsBase< Element, Enable > . . . . .	558
ScalFunctions< Element > . . . . .	553
ScalFunctionsBase< Element, typename enable_if< is_floating_point< Element >::value >::type > . . . . .	559

ScalFunctionsBase< Element, typename enable_if< is_integral< Element >::value >::type > . . . . .	560
Sequential . . . . .	563
Simd128_impl< ArithType, Int, Signed, Size > . . . . .	564
Simd128_impl< true, false, true, 4 > . . . . .	564
Simd128_impl< true, false, true, 8 > . . . . .	564
Simd128i_base . . . . .	622
Simd128_impl< true, true, true, 2 > . . . . .	594
Simd128_impl< true, true, false, 2 > . . . . .	564
Simd128_impl< true, true, true, 4 > . . . . .	604
Simd128_impl< true, true, false, 4 > . . . . .	574
Simd128_impl< true, true, true, 8 > . . . . .	612
Simd128_impl< true, true, false, 8 > . . . . .	584
Simd256_impl< ArithType, Int, Signed, Size > . . . . .	623
Simd256fp_base . . . . .	704
Simd256_impl< true, false, true, 4 > . . . . .	623
Simd256_impl< true, false, true, 8 > . . . . .	624
Simd256i_base . . . . .	705
Simd256_impl< true, true, true, 2 > . . . . .	669
Simd256_impl< true, true, false, 2 > . . . . .	631
Simd256_impl< true, true, true, 4 > . . . . .	678
Simd256_impl< true, true, false, 4 > . . . . .	641
Simd256_impl< true, true, false, 4 > . . . . .	641
Simd256_impl< true, true, true, 8 > . . . . .	695
Simd256_impl< true, true, false, 8 > . . . . .	660
Simd512_impl< ArithType, Int, Signed, Size > . . . . .	705
Simd512_impl< true, false, true, 4 > . . . . .	705
Simd512_impl< true, false, true, 8 > . . . . .	705
Simd512i_base . . . . .	733
Simd256_impl< true, true, true, 4 > . . . . .	678
Simd512_impl< true, true, true, 8 > . . . . .	723
Simd512_impl< true, true, false, 8 > . . . . .	712
SimdChooser< T, bool, bool > . . . . .	734
SimdChooser< T, false, b > . . . . .	734
SimdChooser< T, true, false > . . . . .	734
SimdChooser< T, true, true > . . . . .	735
simdToType< T > . . . . .	735
Single . . . . .	735
Sparse< Field, SparseMatrix_t, IdxT, PtrT > . . . . .	735
Sparse< _Field, SparseMatrix_t::COO > . . . . .	735
Sparse< _Field, SparseMatrix_t::COO_ZO > . . . . .	737
Sparse< _Field, SparseMatrix_t::CSR > . . . . .	739
Sparse< _Field, SparseMatrix_t::CSR_ZO > . . . . .	742
Sparse< _Field, SparseMatrix_t::CSR_HYB > . . . . .	740
Sparse< _Field, SparseMatrix_t::ELL > . . . . .	744
Sparse< _Field, SparseMatrix_t::ELL_ZO > . . . . .	749
Sparse< _Field, SparseMatrix_t::ELL_simd > . . . . .	745
Sparse< _Field, SparseMatrix_t::ELL_simd_ZO > . . . . .	747
Sparse< _Field, SparseMatrix_t::HYB_ZO > . . . . .	750
Sparse< _Field, SparseMatrix_t::SELL > . . . . .	752
Sparse< _Field, SparseMatrix_t::SELL_ZO > . . . . .	754
Sparse< FFPACK::RNSInteger, SparseMatrix_t::COO, int16_t > . . . . .	735
Sparse< FFPACK::RNSInteger, SparseMatrix_t::COO, int32_t > . . . . .	735
Sparse< FFPACK::RNSInteger, SparseMatrix_t::COO, int64_t > . . . . .	735
Sparse< FFPACK::RNSInteger, SparseMatrix_t::COO_ZO, int16_t > . . . . .	735
Sparse< FFPACK::RNSInteger, SparseMatrix_t::COO_ZO, int32_t > . . . . .	735

Sparse< FFPACK::RNSInteger, SparseMatrix_t::COO_ZO, int64_t > . . . . .	735
Sparse< FFPACK::RNSInteger, SparseMatrix_t::CSR, int16_t > . . . . .	735
Sparse< FFPACK::RNSInteger, SparseMatrix_t::CSR, int32_t > . . . . .	735
Sparse< FFPACK::RNSInteger, SparseMatrix_t::CSR, int64_t > . . . . .	735
Sparse< FFPACK::RNSInteger, SparseMatrix_t::CSR_ZO, int16_t > . . . . .	735
Sparse< FFPACK::RNSInteger, SparseMatrix_t::CSR_ZO, int32_t > . . . . .	735
Sparse< FFPACK::RNSInteger, SparseMatrix_t::CSR_ZO, int64_t > . . . . .	735
Sparse< FFPACK::RNSInteger, SparseMatrix_t::ELL, int16_t > . . . . .	735
Sparse< FFPACK::RNSInteger, SparseMatrix_t::ELL, int32_t > . . . . .	735
Sparse< FFPACK::RNSInteger, SparseMatrix_t::ELL, int64_t > . . . . .	735
Sparse< FFPACK::RNSInteger, SparseMatrix_t::ELL_ZO, int16_t > . . . . .	735
Sparse< FFPACK::RNSInteger, SparseMatrix_t::ELL_ZO, int32_t > . . . . .	735
Sparse< FFPACK::RNSInteger, SparseMatrix_t::ELL_ZO, int64_t > . . . . .	735
SpMat< Field, flag > . . . . .	756
StatsMatrix . . . . .	756
Test< Elt > . . . . .	761
TestOneMethod< Simd > . . . . .	764
tfn_minus . . . . .	767
tfn_minus_eq . . . . .	767
tfn_mul . . . . .	768
tfn_mul_eq . . . . .	768
tfn_plus . . . . .	768
tfn_plus_eq . . . . .	769
Threads . . . . .	769
ThreeD . . . . .	769
ThreeDAdaptive . . . . .	769
ThreeDInPlace . . . . .	770
TRSMHelper< ReclterTrait, ParSeqTrait > . . . . .	770
true_type	
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO > > . . . . .	481
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO_ZO > > . . . . .	482
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR > > . . . . .	482
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_HYB > > . . . . .	482
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_ZO > > . . . . .	483
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL > > . . . . .	483
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_ZO > > . . . . .	484
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd > > . . . . .	483
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > > . . . . .	484
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::HYB_ZO > > . . . . .	484
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL > > . . . . .	484
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL_ZO > > . . . . .	485
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO_ZO > > . . . . .	486
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_ZO > > . . . . .	486
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_ZO > > . . . . .	487
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > > . . . . .	486
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL_ZO > > . . . . .	487
support_fast_mod< double > . . . . .	759
support_fast_mod< float > . . . . .	760
support_fast_mod< int64_t > . . . . .	760
TwoD . . . . .	771
TwoDAdaptive . . . . .	771
UnparametricTag . . . . .	771
width< T > . . . . .	772
width< double > . . . . .	772
width< float > . . . . .	772
Winograd . . . . .	772
WinogradPar . . . . .	772



## Chapter 8

# Data Structure Index

### 8.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">AlgoChooser&lt; ModeT, ParSeq &gt;</a>	401
<a href="#">AlgoChooser&lt; ModeCategories::ConvertTo&lt; ElementCategories::RNSElementTag &gt;, ParSeq &gt;</a>	401
<a href="#">ALL&lt; v &gt;</a>	401
<a href="#">ALL&lt; false, v... &gt;</a>	402
<a href="#">ALL&lt; true, v... &gt;</a>	402
<a href="#">ALL&lt;&gt;</a>	402
<a href="#">ArbitraryPrecIntTag</a>	
Arbitrary precision integers: GMP	402
<a href="#">AreEqual&lt; X, Y &gt;</a>	403
<a href="#">AreEqual&lt; X, X &gt;</a>	403
<a href="#">Argument</a>	403
<a href="#">associatedDelayedField&lt; Field &gt;</a>	404
<a href="#">associatedDelayedField&lt; const FFPACK::RNSIntegerMod&lt; RNS &gt; &gt;</a>	404
<a href="#">associatedDelayedField&lt; const Givaro::Modular&lt; T, X &gt; &gt;</a>	405
<a href="#">associatedDelayedField&lt; const Givaro::ModularBalanced&lt; T &gt; &gt;</a>	405
<a href="#">associatedDelayedField&lt; const Givaro::ZRing&lt; T &gt; &gt;</a>	406
<a href="#">Auto</a>	406
<a href="#">Bench&lt; Elt &gt;</a>	406
<a href="#">Bini</a>	409
<a href="#">Block</a>	409
<a href="#">BlockTransposeSIMD&lt; Field, Simd, &gt;</a>	409
<a href="#">callLUdivine_small&lt; Element &gt;</a>	411
<a href="#">callLUdivine_small&lt; double &gt;</a>	411
<a href="#">callLUdivine_small&lt; float &gt;</a>	412
<a href="#">CharpolyFailed</a>	412
<a href="#">Checker_Empty&lt; Field &gt;</a>	412
<a href="#">CheckerImplem_charpoly&lt; Field, Polynomial &gt;</a>	413
<a href="#">CheckerImplem_charpoly&lt; Givaro::ZRing&lt; Givaro::Integer &gt;, Polynomial &gt;</a>	414
<a href="#">CheckerImplem_Det&lt; Field &gt;</a>	415
<a href="#">CheckerImplem_fgemm&lt; Field &gt;</a>	417
<a href="#">CheckerImplem_ftsrsm&lt; Field &gt;</a>	418
<a href="#">CheckerImplem_invert&lt; Field &gt;</a>	419
<a href="#">CheckerImplem_PLUQ&lt; Field &gt;</a>	420
<a href="#">Classic</a>	421
<a href="#">Column</a>	421

CompactElement< Element >	421
CompactElement< double >	421
CompactElement< float >	422
CompactElement< int16_t >	422
CompactElement< int32_t >	422
CompactElement< int64_t >	422
compatible_data_type< Field >	423
compatible_data_type< Givaro::ZRing< double > >	423
compatible_data_type< Givaro::ZRing< float > >	423
Compose< H1, H2 >	424
Simd128_impl< true, true, false, 2 >::Converter	425
Simd128_impl< true, true, false, 4 >::Converter	425
Simd128_impl< true, true, false, 8 >::Converter	425
Simd128_impl< true, true, true, 2 >::Converter	426
Simd128_impl< true, true, true, 4 >::Converter	426
Simd128_impl< true, true, true, 8 >::Converter	426
Simd256_impl< true, false, true, 8 >::Converter	427
Simd256_impl< true, true, false, 2 >::Converter	427
Simd256_impl< true, true, false, 4 >::Converter	427
Simd256_impl< true, true, false, 8 >::Converter	428
Simd256_impl< true, true, true, 2 >::Converter	428
Simd256_impl< true, true, true, 4 >::Converter	428
Simd256_impl< true, true, true, 8 >::Converter	429
Simd512_impl< true, true, false, 8 >::Converter	429
Simd512_impl< true, true, true, 8 >::Converter	429
ConvertTo< T >	
Force conversion to appropriate element type of ElementCategory T	430
Coo< ValT, IdxT >	430
Coo< Field >	432
Coo< ValT, IdxT >	433
CooMat< Field >	435
count_nonconst_lvalue_reference< T >	436
count_nonconst_lvalue_reference< const T &, O... >	436
count_nonconst_lvalue_reference< T &, O... >	436
count_nonconst_lvalue_reference< T, O... >	436
count_nonconst_lvalue_reference<>	437
CsrMat< Field >	437
DefaultBoundedTag	
Use standard field operations, but keeps track of bounds on input and output	438
DefaultTag	
No specific mode of action: use standard field operations	438
DelayedTag	
Performs field operations with delayed mod reductions. Ensures result is reduced	438
DivideAndConquer	438
ElementTraits< Element >	
ElementTraits	438
ElementTraits< double >	439
ElementTraits< FFPACK::rns_double_elt >	439
ElementTraits< float >	439
ElementTraits< Givaro::Integer >	440
ElementTraits< int16_t >	440
ElementTraits< int32_t >	440
ElementTraits< int64_t >	440
ElementTraits< int8_t >	441
ElementTraits< Reclnt::rint< K > >	441
ElementTraits< Reclnt::rmint< K, MG > >	441
ElementTraits< Reclnt::ruint< K > >	442
ElementTraits< uint16_t >	442



ElementTraits< uint32_t > . . . . .	442
ElementTraits< uint64_t > . . . . .	442
ElementTraits< uint8_t > . . . . .	443
ElMat< Field > . . . . .	443
Failure	
A precondition failed . . . . .	444
FailureCharpolyCheck . . . . .	445
FailureDetCheck . . . . .	446
FailureFgemmCheck . . . . .	446
FailureInvertCheck . . . . .	446
FailurePLUQCheck . . . . .	446
FailureTrsmCheck . . . . .	446
FieldSimd< _Field > . . . . .	446
FieldTraits< Field >	
FieldTrait . . . . .	452
FieldTraits< FFPACK::RNSInteger< T > > . . . . .	453
FieldTraits< FFPACK::RNSIntegerMod< T > > . . . . .	453
FieldTraits< Givaro::Modular< Element > > . . . . .	454
FieldTraits< Givaro::ModularBalanced< Element > > . . . . .	454
FieldTraits< Givaro::ZRing< double > > . . . . .	455
FieldTraits< Givaro::ZRing< float > > . . . . .	455
FieldTraits< Givaro::ZRing< Givaro::Integer > > . . . . .	456
FieldTraits< Givaro::ZRing< int16_t > > . . . . .	456
FieldTraits< Givaro::ZRing< int32_t > > . . . . .	457
FieldTraits< Givaro::ZRing< int64_t > > . . . . .	457
FieldTraits< Givaro::ZRing< Reclnt::ruint< K > > > . . . . .	458
FieldTraits< Givaro::ZRing< uint16_t > > . . . . .	458
FieldTraits< Givaro::ZRing< uint32_t > > . . . . .	459
FieldTraits< Givaro::ZRing< uint64_t > > . . . . .	459
Fixed . . . . .	460
FixedPrecIntTag	
Fixed precision integers above machine precision: Givaro::reclnt . . . . .	460
ScalFunctionsBase< Element, typename enable_if< is_floating_point< Element >::value >::type >::FloatingPointTestDistribution	
460	
ForStrategy1D< blocksize_t, Cut, Param > . . . . .	461
ForStrategy2D< blocksize_t, Cut, Param > . . . . .	463
ftmmLeftLowerNoTransNonUnit< Element > . . . . .	467
ftmmLeftLowerNoTransUnit< Element > . . . . .	467
ftmmLeftLowerTransNonUnit< Element > . . . . .	467
ftmmLeftLowerTransUnit< Element > . . . . .	467
ftmmLeftUpperNoTransNonUnit< Element > . . . . .	467
ftmmLeftUpperNoTransUnit< Element > . . . . .	467
ftmmLeftUpperTransNonUnit< Element > . . . . .	467
ftmmLeftUpperTransUnit< Element > . . . . .	468
ftmmRightLowerNoTransNonUnit< Element > . . . . .	468
ftmmRightLowerNoTransUnit< Element > . . . . .	468
ftmmRightLowerTransNonUnit< Element > . . . . .	468
ftmmRightLowerTransUnit< Element > . . . . .	468
ftmmRightUpperNoTransNonUnit< Element > . . . . .	468
ftmmRightUpperNoTransUnit< Element > . . . . .	468
ftmmRightUpperTransNonUnit< Element > . . . . .	468
ftmmRightUpperTransUnit< Element > . . . . .	469
ftsmLeftLowerNoTransNonUnit< Element > . . . . .	469
ftsmLeftLowerNoTransUnit< Element > . . . . .	469
ftsmLeftLowerTransNonUnit< Element > . . . . .	469
ftsmLeftLowerTransUnit< Element > . . . . .	469
ftsmLeftUpperNoTransNonUnit< Element > . . . . .	469
Computes the maximal size for delaying the modular reduction in a triangular system resolution	469

ftsmLeftUpperNoTransUnit< Element > . . . . .	470
ftsmLeftUpperTransNonUnit< Element > . . . . .	470
ftsmLeftUpperTransUnit< Element > . . . . .	470
ftsmRightLowerNoTransNonUnit< Element > . . . . .	470
ftsmRightLowerNoTransUnit< Element > . . . . .	470
ftsmRightLowerTransNonUnit< Element > . . . . .	470
ftsmRightLowerTransUnit< Element > . . . . .	470
ftsmRightUpperNoTransNonUnit< Element > . . . . .	470
ftsmRightUpperNoTransUnit< Element > . . . . .	471
ftsmRightUpperTransNonUnit< Element > . . . . .	471
ftsmRightUpperTransUnit< Element > . . . . .	471
GenericTag	
Default is generic . . . . .	471
GenericTag	
Generic ring . . . . .	471
Grain . . . . .	471
has_minus_eq_impl< C > . . . . .	471
has_minus_impl< C > . . . . .	472
has_mul_eq_impl< C > . . . . .	472
has_mul_impl< C > . . . . .	472
has_operation< T > . . . . .	473
has_plus_eq_impl< C > . . . . .	473
has_plus_impl< C > . . . . .	473
HelperFlag . . . . .	474
HelperMod< Field, ElementTraits > . . . . .	474
HelperMod< Field, ElementCategories::MachineIntTag > . . . . .	475
HelperMod< Field, FFLAS::ElementCategories::ArbitraryPrecIntTag > . . . . .	476
HelperMod< Field, FFLAS::ElementCategories::FixedPrecIntTag > . . . . .	476
HelperMod< Field, FFLAS::ElementCategories::MachineFloatTag > . . . . .	477
Hybrid . . . . .	478
Info . . . . .	478
Info . . . . .	479
is_all_same< Args > . . . . .	480
is_all_same< T, Args... > . . . . .	480
is_all_same<> . . . . .	480
is_simd< T > . . . . .	481
isSparseMatrix< Field, M > . . . . .	481
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO > > . . . . .	481
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO_ZO > > . . . . .	482
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR > > . . . . .	482
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_HYB > > . . . . .	482
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_ZO > > . . . . .	483
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL > > . . . . .	483
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd > > . . . . .	483
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > > . . . . .	484
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_ZO > > . . . . .	484
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::HYB_ZO > > . . . . .	484
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL > > . . . . .	484
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL_ZO > > . . . . .	485
isSparseMatrixMKLFormat< F, M > . . . . .	485
isSparseMatrixSimdFormat< F, M > . . . . .	485
isZOSparseMatrix< F, M > . . . . .	486
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO_ZO > > . . . . .	486
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_ZO > > . . . . .	486
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > > . . . . .	486
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_ZO > > . . . . .	487
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL_ZO > > . . . . .	487
Iterative . . . . .	487

LazyTag	
Performs field operations with delayed mod only when necessary. Result may not be reduced	487
limits< T >	488
limits< char >	488
limits< double >	488
limits< float >	489
limits< Givaro::Integer >	490
limits< int >	490
limits< long >	491
limits< long long >	491
limits< RecInt::rint< K > >	492
limits< RecInt::ruint< K > >	493
limits< short int >	493
limits< signed char >	494
limits< unsigned char >	495
limits< unsigned int >	495
limits< unsigned long >	496
limits< unsigned long long >	496
limits< unsigned short int >	497
MachineFloatTag	
Float or double	498
MachineIntTag	
Short, int, long, long long, and unsigned variants	498
MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >	498
MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >	503
MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >	505
MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< Dest >, ParSeqTrait >	507
MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeqTrait >	509
MMHelper< Field, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >	
FGEMM Helper for Default and ConvertTo modes of operation	511
ModeTraits< Field >	
ModeTraits	512
ModeTraits< Givaro::Modular< Element, Compute > >	513
ModeTraits< Givaro::Modular< Givaro::Integer, Compute > >	513
ModeTraits< Givaro::Modular< int16_t, Compute > >	513
ModeTraits< Givaro::Modular< int32_t, Compute > >	514
ModeTraits< Givaro::Modular< int64_t, uint64_t > >	514
ModeTraits< Givaro::Modular< int8_t, Compute > >	514
ModeTraits< Givaro::Modular< RecInt::ruint< K >, Compute > >	515
ModeTraits< Givaro::Modular< uint16_t, Compute > >	515
ModeTraits< Givaro::Modular< uint32_t, Compute > >	515
ModeTraits< Givaro::Modular< uint8_t, Compute > >	516
ModeTraits< Givaro::ModularBalanced< Element > >	516
ModeTraits< Givaro::ModularBalanced< Givaro::Integer > >	516
ModeTraits< Givaro::ModularBalanced< int16_t > >	517
ModeTraits< Givaro::ModularBalanced< int32_t > >	517
ModeTraits< Givaro::ModularBalanced< int8_t > >	517
ModeTraits< Givaro::Montgomery< T > >	517
ModeTraits< Givaro::ZRing< double > >	518
ModeTraits< Givaro::ZRing< float > >	518
ModeTraits< Givaro::ZRing< Givaro::Integer > >	518
ModularBalanced< T >	519
ModularTag	
This is a modular field like e.g. Modular<T> or ModularBalanced<T>	519
Montgomery< T >	519
need_field_characteristic< Field >	519
need_field_characteristic< Givaro::Modular< Field > >	519

<a href="#">need_field_characteristic&lt; Givaro::ModularBalanced&lt; Field &gt; &gt;</a>	520
<a href="#">NoSimd&lt; T &gt;</a>	520
<a href="#">Parallel&lt; C, P &gt;</a>	522
<a href="#">RNSInteger&lt; RNS &gt;::RandIter</a>	523
<a href="#">RNSIntegerMod&lt; RNS &gt;::RandIter</a>	524
<a href="#">readMyMachineType&lt; Field, T &gt;</a>	525
<a href="#">readMyMachineType&lt; Field, mpz_t &gt;</a>	525
<a href="#">Recursive</a>	526
<a href="#">Recursive</a>	526
<a href="#">rint&lt; K &gt;</a>	526
<a href="#">rns_double</a>	526
<a href="#">rns_double_elt</a>	531
<a href="#">rns_double_elt_cstptr</a>	533
<a href="#">rns_double_elt_ptr</a>	535
<a href="#">rns_double_extended</a>	538
<a href="#">RNSElementTag</a>	
Representation in a Residue Number System	542
<a href="#">RNSInteger&lt; RNS &gt;</a>	542
<a href="#">RNSIntegerMod&lt; RNS &gt;</a>	545
<a href="#">rnsRandIter&lt; RNS &gt;</a>	552
<a href="#">Row</a>	553
<a href="#">ruint&lt; K &gt;</a>	553
<a href="#">ScalFunctions&lt; Element &gt;</a>	553
<a href="#">ScalFunctionsBase&lt; Element, Enable &gt;</a>	558
<a href="#">ScalFunctionsBase&lt; Element, typename enable_if&lt; is_floating_point&lt; Element &gt;::value &gt;::type &gt;</a>	559
<a href="#">ScalFunctionsBase&lt; Element, typename enable_if&lt; is_integral&lt; Element &gt;::value &gt;::type &gt;</a>	560
<a href="#">Sequential</a>	563
<a href="#">Simd128_impl&lt; ArithType, Int, Signed, Size &gt;</a>	564
<a href="#">Simd128_impl&lt; true, false, true, 4 &gt;</a>	564
<a href="#">Simd128_impl&lt; true, false, true, 8 &gt;</a>	564
<a href="#">Simd128_impl&lt; true, true, false, 2 &gt;</a>	564
<a href="#">Simd128_impl&lt; true, true, false, 4 &gt;</a>	574
<a href="#">Simd128_impl&lt; true, true, false, 8 &gt;</a>	584
<a href="#">Simd128_impl&lt; true, true, true, 2 &gt;</a>	594
<a href="#">Simd128_impl&lt; true, true, true, 4 &gt;</a>	604
<a href="#">Simd128_impl&lt; true, true, true, 8 &gt;</a>	612
<a href="#">Simd128i_base</a>	622
<a href="#">Simd256_impl&lt; ArithType, Int, Signed, Size &gt;</a>	623
<a href="#">Simd256_impl&lt; true, false, true, 4 &gt;</a>	623
<a href="#">Simd256_impl&lt; true, false, true, 8 &gt;</a>	624
<a href="#">Simd256_impl&lt; true, true, false, 2 &gt;</a>	631
<a href="#">Simd256_impl&lt; true, true, false, 4 &gt;</a>	641
<a href="#">Simd256_impl&lt; true, true, false, 8 &gt;</a>	660
<a href="#">Simd256_impl&lt; true, true, true, 2 &gt;</a>	669
<a href="#">Simd256_impl&lt; true, true, true, 4 &gt;</a>	678
<a href="#">Simd256_impl&lt; true, true, true, 8 &gt;</a>	695
<a href="#">Simd256fp_base</a>	704
<a href="#">Simd256i_base</a>	705
<a href="#">Simd512_impl&lt; ArithType, Int, Signed, Size &gt;</a>	705
<a href="#">Simd512_impl&lt; true, false, true, 4 &gt;</a>	705
<a href="#">Simd512_impl&lt; true, false, true, 8 &gt;</a>	705
<a href="#">Simd512_impl&lt; true, true, false, 8 &gt;</a>	712
<a href="#">Simd512_impl&lt; true, true, true, 8 &gt;</a>	723
<a href="#">Simd512i_base</a>	733
<a href="#">SimdChooser&lt; T, bool, bool &gt;</a>	734
<a href="#">SimdChooser&lt; T, false, b &gt;</a>	734
<a href="#">SimdChooser&lt; T, true, false &gt;</a>	734
<a href="#">SimdChooser&lt; T, true, true &gt;</a>	735

<code>simdToType&lt; T &gt;</code>	735
<code>Single</code>	735
<code>Sparse&lt; Field, SparseMatrix_t, IdxT, PtrT &gt;</code>	735
<code>Sparse&lt; _Field, SparseMatrix_t::COO &gt;</code>	735
<code>Sparse&lt; _Field, SparseMatrix_t::COO_ZO &gt;</code>	737
<code>Sparse&lt; _Field, SparseMatrix_t::CSR &gt;</code>	739
<code>Sparse&lt; _Field, SparseMatrix_t::CSR_HYB &gt;</code>	740
<code>Sparse&lt; _Field, SparseMatrix_t::CSR_ZO &gt;</code>	742
<code>Sparse&lt; _Field, SparseMatrix_t::ELL &gt;</code>	744
<code>Sparse&lt; _Field, SparseMatrix_t::ELL_simd &gt;</code>	745
<code>Sparse&lt; _Field, SparseMatrix_t::ELL_simd_ZO &gt;</code>	747
<code>Sparse&lt; _Field, SparseMatrix_t::ELL_ZO &gt;</code>	749
<code>Sparse&lt; _Field, SparseMatrix_t::HYB_ZO &gt;</code>	750
<code>Sparse&lt; _Field, SparseMatrix_t::SELL &gt;</code>	752
<code>Sparse&lt; _Field, SparseMatrix_t::SELL_ZO &gt;</code>	754
<code>SpMat&lt; Field, flag &gt;</code>	756
<code>StatsMatrix</code>	756
<code>support_fast_mod&lt; T &gt;</code>	759
<code>support_fast_mod&lt; double &gt;</code>	759
<code>support_fast_mod&lt; float &gt;</code>	760
<code>support_fast_mod&lt; int64_t &gt;</code>	760
<code>support_simd&lt; T &gt;</code>	760
<code>support_simd_add&lt; T &gt;</code>	761
<code>support_simd_mod&lt; T &gt;</code>	761
<code>Test&lt; Elt &gt;</code>	761
<code>TestOneMethod&lt; Simd &gt;</code>	764
<code>tfn_minus</code>	767
<code>tfn_minus_eq</code>	767
<code>tfn_mul</code>	768
<code>tfn_mul_eq</code>	768
<code>tfn_plus</code>	768
<code>tfn_plus_eq</code>	769
<code>Threads</code>	769
<code>ThreeD</code>	769
<code>ThreeDAdaptive</code>	769
<code>ThreeDInPlace</code>	770
<code>TRSMHelper&lt; ReclterTrait, ParSeqTrait &gt;</code>	
<code>TRSM Helper</code>	770
<code>TwoD</code>	771
<code>TwoDAdaptive</code>	771
<code>UnparametricTag</code>	
If the field uses a representation with infix operators	771
<code>width&lt; T &gt;</code>	772
<code>width&lt; double &gt;</code>	772
<code>width&lt; float &gt;</code>	772
<code>Winograd</code>	772
<code>WinogradPar</code>	772



# Chapter 9

## File Index

### 9.1 File List

Here is a list of all files with brief descriptions:

<a href="#">arithprog.C</a>	773
<a href="#">autotune/charpoly.C</a>	773
<a href="#">examples/charpoly.C</a>	774
<a href="#">fsyrk.C</a>	775
<a href="#">fsytrf.C</a>	775
<a href="#">ftrtri.C</a>	776
<a href="#">autotune/pluq.C</a>	777
<a href="#">examples/pluq.C</a>	778
<a href="#">winograd.C</a>	778
<a href="#">benchmark-charpoly-mp.C</a>	779
<a href="#">benchmark-charpoly.C</a>	779
<a href="#">benchmark-checkers.C</a>	780
<a href="#">benchmark-dgemm.C</a>	781
<a href="#">benchmark-dgetrf.C</a>	782
<a href="#">benchmark-dgetri.C</a>	783
<a href="#">benchmark-dsytrf.C</a>	783
<a href="#">benchmark-dtrsm.C</a>	784
<a href="#">benchmark-dtrtri.C</a>	784
<a href="#">benchmark-fadd-lvl2.C</a>	785
<a href="#">benchmark-fdot.C</a>	785
<a href="#">benchmark-fgemm-mp.C</a>	786
<a href="#">benchmark-fgemm-rns.C</a>	787
<a href="#">benchmark-fgemm.C</a>	789
<a href="#">benchmark-fgemv-mp.C</a>	789
<a href="#">benchmark-fgemv.C</a>	791
<a href="#">benchmark-fgesv.C</a>	794
<a href="#">benchmark-fsyr2k.C</a>	794
<a href="#">benchmark-fsyrk.C</a>	795
<a href="#">benchmark-fsytrf.C</a>	795
<a href="#">benchmark-ftrsm-mp.C</a>	796
<a href="#">benchmark-ftrsm.C</a>	797
<a href="#">benchmark-ftrsv.C</a>	797
<a href="#">benchmark-ftrtri.C</a>	798
<a href="#">benchmark-inverse.C</a>	798
<a href="#">benchmark-lqup-mp.C</a>	799

benchmark-lqup.C	800
benchmark-pluq.C	800
benchmark-quasisep.C	801
benchmark-storage-transpose.C	802
benchmark-wino.C	803
mainpage.doxy	804
det.C	804
matmul.C	804
rank.C	805
solve.C	805
checker_charpoly.inl	805
checker_det.inl	806
checker_empty.h	806
checker_fgemm.inl	806
checker_ftrsm.inl	807
checker_invert.inl	807
checker_pluq.inl	807
checkers.doxy	808
checkers_fflas.h	808
checkers_fflas.inl	808
checkers_ffpack.h	809
checkers_ffpack.inl	809
config-blas.h	810
config.h	817
fflas-ffpack/config.h	820
fflas-ffpack-config.h	
Defaults for optimised values	823
fflas-ffpack-default-thresholds.h	824
fflas-ffpack-thresholds.h	825
fflas-ffpack.doxy	825
fflas-ffpack.h	
Includes FFLAS and FFPACK	825
fflas.doxy	825
fflas.h	
Finite Field Linear Algebra Subroutines	825
fflas_bounds.inl	827
fflas_enum.h	827
fflas_fadd.h	828
fflas_fadd.inl	829
fflas_fassign.h	830
fflas_fassign.inl	831
fflas_faxpy.inl	831
fflas_fdot.inl	832
fflas_fgemm.inl	833
fgemm_classical.inl	835
fgemm_classical_mp.inl	
Matrix multiplication with multiprecision input (either over $\mathbb{Z}$ or over $\mathbb{Z}/p\mathbb{Z}$ )	835
fgemm_winograd.inl	837
matmul.doxy	839
schedule_bini.inl	
Bini implementation	839
schedule_winograd.inl	839
schedule_winograd_acc.inl	840
schedule_winograd_acc_ip.inl	841
schedule_winograd_ip.inl	841
fflas_fgmv.inl	842
fflas_fgmv_mp.inl	844
fflas_fger.inl	844



fflas_fger_mp.inl	845
fflas_freduce.h	846
fflas_freduce.inl	847
fflas_freduce_mp.inl	849
fflas_freivalds.inl	849
fflas_fscal.h	850
fflas_fscal.inl	850
fflas_fscal_mp.inl	851
fflas_fsyr2k.inl	852
fflas_fsyrk.inl	853
fflas_fsyrk_strassen.inl	854
fflas_ftrmm.inl	855
fflas_ftrsm.inl	856
fflas_ftrsm_mp.inl	
Triangular system with matrix right hand side over multiprecision domain (either over $\mathbb{Z}$ or over $\mathbb{Z}/p\mathbb{Z}$ )	857
fflas_ftrsv.inl	857
fflas_helpers.inl	858
igemm.doxy	859
igemm.h	859
igemm.inl	859
igemm_kernels.h	860
igemm_kernels.inl	861
igemm_tools.h	861
igemm_tools.inl	862
fflas_level1.inl	862
fflas_level2.inl	865
fflas_level3.inl	867
fflas_pfgemm.inl	870
fflas_pftrsm.inl	871
fflas_simd.h	871
simd.doxy	873
simd128.inl	873
simd128_double.inl	874
simd128_float.inl	874
simd128_int16.inl	874
simd128_int32.inl	875
simd128_int64.inl	875
simd256.inl	876
simd256_double.inl	876
simd256_float.inl	877
simd256_int16.inl	877
simd256_int32.inl	878
simd256_int64.inl	878
simd512.inl	879
simd512_double.inl	879
simd512_float.inl	880
simd512_int32.inl	880
simd512_int64.inl	881
simd_modular.inl	881
fflas_sparse.h	881
fflas_sparse.inl	886
coo.h	888
coo_spmv.inl	888
coo_spmv.inl	889
coo_utils.inl	890
csr.h	890
csr_pspmm.inl	891

csr_pspmv.inl	892
csr_spmm.inl	892
csr_spmv.inl	894
csr_utils.inl	894
csr_hyb.h	895
csr_hyb_pspmm.inl	895
csr_hyb_pspmv.inl	896
csr_hyb_spmm.inl	897
csr_hyb_spmv.inl	897
csr_hyb_utils.inl	898
ell.h	898
ell_pspmm.inl	899
ell_pspmv.inl	899
ell_spmm.inl	900
ell_spmv.inl	901
ell_utils.inl	902
ell_simd.h	902
ell_simd_pspmv.inl	903
ell_simd_spmv.inl	904
ell_simd_utils.inl	905
hyb_zo.h	905
hyb_zo_pspmm.inl	905
hyb_zo_pspmv.inl	906
hyb_zo_spmm.inl	906
hyb_zo_spmv.inl	907
hyb_zo_utils.inl	907
read_sparse.h	908
sell.h	909
sell_pspmv.inl	909
sell_spmv.inl	910
sell_utils.inl	911
sparse_matrix_traits.h	912
utils.h	913
fflas_transpose.h	
Transpose the storage of the matrix (switch between row and col major mode)	913
ffpack.dox	914
ffpack.h	
Set of elimination based routines for dense linear algebra	914
ffpack.inl	924
ffpack_bruhatgen.inl	925
ffpack_charpoly.inl	926
ffpack_charpoly_danilevski.inl	927
ffpack_charpoly_kgfast.inl	927
ffpack_charpoly_kgfastgeneralized.inl	928
ffpack_charpoly_kglu.inl	928
ffpack_charpoly_mp.inl	929
ffpack_det_mp.inl	929
ffpack_echelonforms.inl	930
ffpack_fgesv.inl	931
ffpack_fgetrs.inl	932
ffpack_frobenius.inl	932
ffpack_fsytrf.inl	933
ffpack_ftrssyr2k.inl	934
ffpack_ftrstr.inl	935
ffpack_fttr.inl	935
ffpack_invert.inl	936
ffpack_krylovelim.inl	937
ffpack_ludivine.inl	937

ffpack_ludivine_mp.inl	938
ffpack_minpoly.inl	938
ffpack_permutation.inl	939
ffpack_pluq.inl	941
ffpack_pluq_mp.inl	942
ffpack_ppluq.inl	942
ffpack_rankprofiles.inl	943
field-traits.h	
Field Traits	944
field.doxy	947
rns-double-elt.h	
Rns elt structure with double support	947
rns-double-recint.inl	947
rns-double.h	
Rns structure with double support	948
rns-double.inl	949
rns-integer-mod.h	
Representation of $\mathbb{Z}/p\mathbb{Z}$ using RNS representation (note: fixed precision)	949
rns-integer.h	
Representation of $\mathbb{Z}$ using RNS representation (note: fixed precision)	950
rns.h	950
rns.inl	951
interfaces.doxy	951
fflas_c.h	951
fflas_L1_inst.C	963
fflas_L1_inst.h	964
fflas_L1_inst_implem.inl	965
fflas_L2_inst.C	966
fflas_L2_inst.h	967
fflas_L2_inst_implem.inl	968
fflas_L3_inst.C	970
fflas_L3_inst.h	971
fflas_L3_inst_implem.inl	972
fflas_lv1.C	
C functions calls for level 1 <b>FFLAS</b> in flas-c.h	973
fflas_lv2.C	
C functions calls for level 2 <b>FFLAS</b> in flas-c.h	977
fflas_lv3.C	
C functions calls for level 3 <b>FFLAS</b> in flas-c.h	982
fflas_sparse.C	
C functions calls for level 1.5 and 2.5 <b>FFLAS</b> in flas-c.h	984
ffpack.C	
C functions calls for <b>FFPACK</b> in ffpack-c.h	984
ffpack_c.h	1005
ffpack_inst.C	1024
ffpack_inst.h	1025
ffpack_inst_implem.inl	1027
blockcuts.inl	1030
fflas_plevel1.h	1031
kaapi_routines.inl	1032
parallel.h	1032
pfgemm_variants.inl	1039
pfgemv.inl	1040
align-allocator.h	1040
args-parser.h	1040
bit_manipulation.h	1042
cast.h	1043

debug.h	
Various utilities for debugging	1043
fflas_intrinsic.h	1044
fflas_io.h	1044
fflas_memory.h	1045
fflas_randommatrix.h	1046
flimits.h	1048
Matio.h	1049
test-utils.h	1049
timer.h	1050
cblas.C	1050
clapack.C	1051
cuda.C	1052
fblas.C	1052
lapack.C	1053
regression-check.C	1053
test-charpoly-check.C	1054
test-charpoly.C	1055
test-compressQ.C	1056
test-det-check.C	1057
test-det.C	1057
test-echelon.C	1058
test-fadd.C	1060
test-fdot.C	1061
test-fgemm-check.C	1063
test-fgemm.C	1064
test-fgemv.C	1066
test-fger.C	1068
test-fgesv.C	1070
test-finit.C	1071
test-fscal.C	1072
test-fsyr2k.C	1073
test-fsyrrk.C	1074
test-fsytrf.C	1076
test-ftrmm.C	1078
test-ftrmv.C	1079
test-ftrsm-check.C	1080
test-ftrsm.C	1081
test-ftrssyr2k.C	1082
test-ftrstr.C	1083
test-ftrsv.C	1084
test-ftrtri.C	1085
test-interfaces-c.c	1086
test-invert-check.C	1087
test-io.C	1087
test-lu.C	1088
test-maxdelayeddim.C	1092
test-minpoly.C	1093
test-multifile1.C	1093
test-multifile2.C	1094
test-nullspace.C	1094
test-permutations.C	1095
test-pluq-check.C	1096
test-quasisep.C	1097
test-rankprofiles.C	1098
test-rpm.C	1099
test-simd.C	1100
test-solve.C	1103

test-storage-transpose.C	1104
101-fgemm.C	1104
2x2-fgemm.C	1105
2x2-ftrsv.C	1105
2x2-pluq.C	1106
fflas-101_1.C	1106
fflas-101_3.C	1106
fflas_101.C	1107
fflas_101_lvl1.C	1107
ffpack-fgesv.C	1107
ffpack-solve.C	1108



# Chapter 10

## Topic Documentation

### 10.1 CHECKER

Class CHECKER provides functions to verify computations in [FFLAS](#) and [FFPACK](#).

### 10.2 FFLAS-FFPACK

the [FFLAS FFPACK](#) library

#### Topics

- [FFLAS](#)
- [Interfaces](#)

#### 10.2.1 Detailed Description

the [FFLAS FFPACK](#) library

C++ header library for fast exact dense linear algebra

See also

[FFLAS](#)  
[FFPACK](#)

#### 10.2.2 FFLAS

The C-style wrapper of BLAS for finite field linear algebra.

[FFLAS](#), Finite Field Linear Algebra Subroutines, provide basic linear algebra subroutines based on the BLAS interface. Therefore, the specifications are in C style; only the field given as a template parameter requires C++.

As much as possible, these routines use [ATLAS/BLAS](#) computations and achieve therefore high efficiency.

### 10.2.3 Interfaces

Intefaces for FFLAS-FFPACK

C interface in folder

See also

libs

## 10.3 Matrix Multiplication Algorithms

### Files

- file [schedule\\_bini.inl](#)  
*Bini implementation.*

### 10.3.1 Detailed Description

Matrix Multiplication (level 3) algorithms

**Todo** biblio

## 10.4 SIMD wrapper

wraps SIMD functions Support SSE4.1, AVX, AVX2.

**Todo** biblio

## 10.5 FFPACK

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### 10.5.1 Detailed Description

Class [FFPACK](#) provides functions using fflas much as Lapack uses BLAS.



## 10.6 FFLAS-FFPACK fields

### Files

- file [rns-double-elt.h](#)  
*rns elt structure with double support*
- file [rns-double.h](#)  
*rns structure with double support*
- file [rns-integer-mod.h](#)  
*representation of  $\mathbb{Z}/p\mathbb{Z}$  using RNS representation (note: fixed precision)*
- file [rns-integer.h](#)  
*representation of  $\mathbb{Z}$  using RNS representation (note: fixed precision)*
- file [rns.h](#)

### 10.6.1 Detailed Description

fields in the FFLAS-FFPACK library

Unparametric/Random elements

[Todo](#) biblio

## 10.7 RNS

just include them all

just include them all



# Chapter 11

## Namespace Documentation

### 11.1 FFLAS Namespace Reference

#### Namespaces

- namespace [\\_frtranspose\\_impl](#)
- namespace [BLAS3](#)
- namespace [csr\\_hyb\\_details](#)
- namespace [CuttingStrategy](#)
- namespace [details](#)
- namespace [details\\_spmv](#)
- namespace [ElementCategories](#)
- namespace [FieldCategories](#)

*Traits and categories will need to be placed in a proper file later.*

- namespace [MMHelperAlgo](#)
- namespace [ModeCategories](#)

*Specifies the mode of action for an algorithm w.r.t.*

- namespace [ParSeqHelper](#)

*ParSeqHelper for both fgemm and ftrsm.*

- namespace [Protected](#)
- namespace [sell\\_details](#)
- namespace [sparse\\_details](#)
- namespace [sparse\\_details\\_impl](#)
- namespace [StrategyParameter](#)
- namespace [StructureHelper](#)

*StructureHelper for ftrsm.*

- namespace [vectorised](#)

#### Data Structures

- struct [AlgoChooser](#)
- struct [AlgoChooser< ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq >](#)
- struct [associatedDelayedField](#)
- struct [associatedDelayedField< const FFPACK::RNSIntegerMod< RNS > >](#)
- struct [associatedDelayedField< const Givaro::Modular< T, X > >](#)
- struct [associatedDelayedField< const Givaro::ModularBalanced< T > >](#)
- struct [associatedDelayedField< const Givaro::ZRing< T > >](#)

- struct [BlockTransposeSIMD](#)
- struct [Checker\\_Empty](#)
- class [CheckerImplem\\_fgemm](#)
- class [CheckerImplem\\_ftsm](#)
- struct [CooMat](#)
- struct [CsrMat](#)
- struct [ElementTraits](#)
  - ElementTraits.*
- struct [ElementTraits< double >](#)
- struct [ElementTraits< FFPACK::rns\\_double\\_elt >](#)
- struct [ElementTraits< float >](#)
- struct [ElementTraits< Givaro::Integer >](#)
- struct [ElementTraits< int16\\_t >](#)
- struct [ElementTraits< int32\\_t >](#)
- struct [ElementTraits< int64\\_t >](#)
- struct [ElementTraits< int8\\_t >](#)
- struct [ElementTraits< Reclnt::rint< K > >](#)
- struct [ElementTraits< Reclnt::rmint< K, MG > >](#)
- struct [ElementTraits< Reclnt::ruint< K > >](#)
- struct [ElementTraits< uint16\\_t >](#)
- struct [ElementTraits< uint32\\_t >](#)
- struct [ElementTraits< uint64\\_t >](#)
- struct [ElementTraits< uint8\\_t >](#)
- struct [ElIMat](#)
- struct [FieldTraits](#)
  - FieldTrait.*
- struct [FieldTraits< FFPACK::RNSInteger< T > >](#)
- struct [FieldTraits< FFPACK::RNSIntegerMod< T > >](#)
- struct [FieldTraits< Givaro::Modular< Element > >](#)
- struct [FieldTraits< Givaro::ModularBalanced< Element > >](#)
- struct [FieldTraits< Givaro::ZRing< double > >](#)
- struct [FieldTraits< Givaro::ZRing< float > >](#)
- struct [FieldTraits< Givaro::ZRing< Givaro::Integer > >](#)
- struct [FieldTraits< Givaro::ZRing< int16\\_t > >](#)
- struct [FieldTraits< Givaro::ZRing< int32\\_t > >](#)
- struct [FieldTraits< Givaro::ZRing< int64\\_t > >](#)
- struct [FieldTraits< Givaro::ZRing< Reclnt::ruint< K > > >](#)
- struct [FieldTraits< Givaro::ZRing< uint16\\_t > >](#)
- struct [FieldTraits< Givaro::ZRing< uint32\\_t > >](#)
- struct [FieldTraits< Givaro::ZRing< uint64\\_t > >](#)
- struct [ForStrategy1D](#)
- struct [ForStrategy2D](#)
- struct [has\\_minus\\_eq\\_impl](#)
- struct [has\\_minus\\_impl](#)
- struct [has\\_mul\\_eq\\_impl](#)
- struct [has\\_mul\\_impl](#)
- struct [has\\_operation](#)
- struct [has\\_plus\\_eq\\_impl](#)
- struct [has\\_plus\\_impl](#)
- struct [HelperFlag](#)
- struct [isSparseMatrix](#)
- struct [isSparseMatrix< Field, Sparse< Field, SparseMatrix\\_t::COO > >](#)
- struct [isSparseMatrix< Field, Sparse< Field, SparseMatrix\\_t::COO\\_ZO > >](#)
- struct [isSparseMatrix< Field, Sparse< Field, SparseMatrix\\_t::CSR > >](#)

- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::CSR\_HYB > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::CSR\_ZO > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::ELL > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::ELL\_simd > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::ELL\_simd\_ZO > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::ELL\_ZO > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::HYB\_ZO > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::SELL > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::SELL\_ZO > >
- struct [isSparseMatrixMKLFormat](#)
- struct [isSparseMatrixSimdFormat](#)
- struct [isZOSparseMatrix](#)
- struct [isZOSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::COO\_ZO > >
- struct [isZOSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::CSR\_ZO > >
- struct [isZOSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::ELL\_simd\_ZO > >
- struct [isZOSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::ELL\_ZO > >
- struct [isZOSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::SELL\_ZO > >
- struct [MMHelper](#)
- struct [MMHelper](#)< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >
- struct [MMHelper](#)< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >
- struct [MMHelper](#)< Field, AlgoTrait, ModeCategories::ConvertTo< Dest >, ParSeqTrait >
- struct [MMHelper](#)< Field, AlgoTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeqTrait >
- struct [MMHelper](#)< Field, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >
- *FGEMM Helper for Default and ConvertTo modes of operation.*
- struct [ModeTraits](#)
- *ModeTraits.*
- struct [ModeTraits](#)< Givaro::Modular< Element, Compute > >
- struct [ModeTraits](#)< Givaro::Modular< Givaro::Integer, Compute > >
- struct [ModeTraits](#)< Givaro::Modular< int16\_t, Compute > >
- struct [ModeTraits](#)< Givaro::Modular< int32\_t, Compute > >
- struct [ModeTraits](#)< Givaro::Modular< int64\_t, uint64\_t > >
- struct [ModeTraits](#)< Givaro::Modular< int8\_t, Compute > >
- struct [ModeTraits](#)< Givaro::Modular< RecInt::ruint< K >, Compute > >
- struct [ModeTraits](#)< Givaro::Modular< uint16\_t, Compute > >
- struct [ModeTraits](#)< Givaro::Modular< uint32\_t, Compute > >
- struct [ModeTraits](#)< Givaro::Modular< uint8\_t, Compute > >
- struct [ModeTraits](#)< Givaro::ModularBalanced< Element > >
- struct [ModeTraits](#)< Givaro::ModularBalanced< Givaro::Integer > >
- struct [ModeTraits](#)< Givaro::ModularBalanced< int16\_t > >
- struct [ModeTraits](#)< Givaro::ModularBalanced< int32\_t > >
- struct [ModeTraits](#)< Givaro::ModularBalanced< int8\_t > >
- struct [ModeTraits](#)< Givaro::Montgomery< T > >
- struct [ModeTraits](#)< Givaro::ZRing< double > >
- struct [ModeTraits](#)< Givaro::ZRing< float > >
- struct [ModeTraits](#)< Givaro::ZRing< Givaro::Integer > >
- struct [readMyMachineType](#)
- struct [readMyMachineType](#)< Field, mpz\_t >
- struct [Sparse](#)
- struct [Sparse](#)< \_Field, SparseMatrix\_t::COO >
- struct [Sparse](#)< \_Field, SparseMatrix\_t::COO\_ZO >
- struct [Sparse](#)< \_Field, SparseMatrix\_t::CSR >
- struct [Sparse](#)< \_Field, SparseMatrix\_t::CSR\_HYB >
- struct [Sparse](#)< \_Field, SparseMatrix\_t::CSR\_ZO >
- struct [Sparse](#)< \_Field, SparseMatrix\_t::ELL >

- struct [Sparse](#)< [\\_Field](#), [SparseMatrix\\_t::ELL\\_simd](#) >
- struct [Sparse](#)< [\\_Field](#), [SparseMatrix\\_t::ELL\\_simd\\_ZO](#) >
- struct [Sparse](#)< [\\_Field](#), [SparseMatrix\\_t::ELL\\_ZO](#) >
- struct [Sparse](#)< [\\_Field](#), [SparseMatrix\\_t::HYB\\_ZO](#) >
- struct [Sparse](#)< [\\_Field](#), [SparseMatrix\\_t::SELL](#) >
- struct [Sparse](#)< [\\_Field](#), [SparseMatrix\\_t::SELL\\_ZO](#) >
- struct [SpMat](#)
- struct [StatsMatrix](#)
- struct [support\\_fast\\_mod](#)
- struct [support\\_fast\\_mod](#)< double >
- struct [support\\_fast\\_mod](#)< float >
- struct [support\\_fast\\_mod](#)< int64\_t >
- struct [support\\_simd](#)
- struct [support\\_simd\\_add](#)
- struct [support\\_simd\\_mod](#)
- struct [tfn\\_minus](#)
- struct [tfn\\_minus\\_eq](#)
- struct [tfn\\_mul](#)
- struct [tfn\\_mul\\_eq](#)
- struct [tfn\\_plus](#)
- struct [tfn\\_plus\\_eq](#)
- struct [TRSMHelper](#)

*TRSM Helper.*

## Typedefs

- template<class [Field](#) >  
using [Checker\\_fgemm](#) = [FFLAS::Checker\\_Empty](#)<[Field](#)>
- template<class [Field](#) >  
using [Checker\\_ftrsm](#) = [FFLAS::Checker\\_Empty](#)<[Field](#)>
- template<class [Field](#) >  
using [ForceCheck\\_fgemm](#) = [CheckerImplem\\_fgemm](#)<[Field](#)>
- template<class [Field](#) >  
using [ForceCheck\\_ftrsm](#) = [CheckerImplem\\_ftrsm](#)<[Field](#)>
- using [ZOSparseMatrix](#) = std::true\_type
- using [NotZOSparseMatrix](#) = std::false\_type
- using [SimdSparseMatrix](#) = std::true\_type
- using [NoSimdSparseMatrix](#) = std::false\_type
- using [MKLSparseMatrixFormat](#) = std::true\_type
- using [NotMKLSparseMatrixFormat](#) = std::false\_type
- template<class T >  
using [has\\_plus](#) = typename std::conditional<std::is\_arithmetic<T>::value, std::true\_type, [has\\_plus\\_impl](#)<T>>::type
- template<class T >  
using [has\\_minus](#) = typename std::conditional<std::is\_arithmetic<T>::value, std::true\_type, [has\\_minus\\_impl](#)<T>>::type
- template<class T >  
using [has\\_equal](#) = typename std::conditional<std::is\_arithmetic<T>::value, std::true\_type, std::is\_copy\_assignable<T>>::type
- template<class T >  
using [has\\_plus\\_eq](#) = typename std::conditional<std::is\_arithmetic<T>::value, std::true\_type, [has\\_plus\\_eq\\_impl](#)<T>>::type
- template<class T >  
using [has\\_minus\\_eq](#) = typename std::conditional<std::is\_arithmetic<T>::value, std::true\_type, [has\\_minus\\_eq\\_impl](#)<T>>::type

- `template<class T >`  
using `has_mul` = `typename std::conditional<std::is_arithmetic<T>::value, std::true_type, has_mul_impl<T>>>::type`
- `template<class T >`  
using `has_mul_eq` = `typename std::conditional<std::is_arithmetic<T>::value, std::true_type, has_mul_eq_impl<T>>>::type`
- `typedef Givaro::Timer` `Timer`
- `typedef Givaro::BaseTimer` `BaseTimer`
- `typedef Givaro::UserTimer` `UserTimer`
- `typedef Givaro::SysTimer` `SysTimer`

## Enumerations

- enum `FFLAS_ORDER` { `FflasRowMajor` = 101 , `FflasColMajor` = 102 }  
*Storage by row or col ?*
- enum `FFLAS_TRANSPOSE` { `FflasNoTrans` = 111 , `FflasTrans` = 112 }  
*Is matrix transposed ?*
- enum `FFLAS_UPLO` { `FflasUpper` = 121 , `FflasLower` = 122 , `FflasLeftTri` = 123 , `FflasRightTri` = 124 }  
*Is triangular matrix's shape upper ?*
- enum `FFLAS_DIAG` { `FflasNonUnit` = 131 , `FflasUnit` = 132 }  
*Is the triangular matrix implicitly unit diagonal ?*
- enum `FFLAS_SIDE` { `FflasLeft` = 141 , `FflasRight` = 142 }  
*On what side ?*
- enum `FFLAS_BASE` { `FflasDouble` = 151 , `FflasFloat` = 152 , `FflasGeneric` = 153 }  
*FFLAS\_BASE determines the type of the element representation for Matrix Mult kernel.*
- enum `number_kind` { `zero` = 0 , `one` = 1 , `mone` = -1 , `other` = 2 }
- enum class `SparseMatrix_t` {  
`CSR` , `CSR_ZO` , `CSC` , `CSC_ZO` ,  
`COO` , `COO_ZO` , `ELL` , `ELL_ZO` ,  
`SELL` , `SELL_ZO` , `ELL_simd` , `ELL_simd_ZO` ,  
`CSR_HYB` , `HYB_ZO` }
- enum `FFLAS_FORMAT` {  
`FflasAuto` = 0 , `FflasDense` = 1 , `FflasSMS` = 2 , `FflasBinary` = 3 ,  
`FflasMath` = 4 , `FflasMaple` = 5 , `FflasSageMath` = 6 }

## Functions

- `Givaro::Integer` `InfNorm` (`const size_t M`, `const size_t N`, `const Givaro::Integer *A`, `const size_t lda`)
- `template<class T >`  
`const T & min3` (`const T &m`, `const T &n`, `const T &k`)
- `template<class T >`  
`const T & max3` (`const T &m`, `const T &n`, `const T &k`)
- `template<class T >`  
`const T & min4` (`const T &m`, `const T &n`, `const T &k`, `const T &l`)
- `template<class T >`  
`const T & max4` (`const T &m`, `const T &n`, `const T &k`, `const T &l`)
- `template<class Field >`  
void `fadd` (`const Field &F`, `const size_t N`, `typename Field::ConstElement_ptr A`, `const size_t inca`, `typename Field::ConstElement_ptr B`, `const size_t incb`, `typename Field::Element_ptr C`, `const size_t incc`)
- `template<class Field >`  
void `faddin` (`const Field &F`, `const size_t N`, `typename Field::ConstElement_ptr B`, `const size_t incb`, `typename Field::Element_ptr C`, `const size_t incc`)

- `template<class Field >`  
`void fsub (const Field &F, const size_t N, typename Field::ConstElement_ptr A, const size_t inca, typename Field::ConstElement_ptr B, const size_t incb, typename Field::Element_ptr C, const size_t incc)`
- `template<class Field >`  
`void fsubin (const Field &F, const size_t N, typename Field::ConstElement_ptr B, const size_t incb, typename Field::Element_ptr C, const size_t incc)`
- `template<class Field >`  
`void fadd (const Field &F, const size_t N, typename Field::ConstElement_ptr A, const size_t inca, const typename Field::Element alpha, typename Field::ConstElement_ptr B, const size_t incb, typename Field::Element_ptr C, const size_t incc)`
- `template<class Field >`  
`void pfadd (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc, const size_t numths)`
- `template<class Field >`  
`void pfsub (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc, const size_t numths)`
- `template<class Field >`  
`void pfaddin (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc, size_t numths)`
- `template<class Field >`  
`void pfsubin (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc, size_t numths)`
- `template<class Field >`  
`void fadd (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc)`  
*fadd : matrix addition.*
- `template<class Field >`  
`void fsub (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc)`  
*fsub : matrix subtraction.*
- `template<class Field >`  
`void faddin (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc)`  
*faddin*
- `template<class Field >`  
`void faddin (const Field &F, const FFLAS_UPLO uplo, const size_t N, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc)`  
*fadding for symmetric matrices*
- `template<class Field >`  
`void fsubin (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc)`  
*fsubin  $C = C - B$*
- `template<class Field >`  
`void fadd (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element alpha, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc)`  
*fadd : matrix addition with scaling.*
- `template<class Field >`  
`void fassign (const Field &F, const size_t N, typename Field::ConstElement_ptr Y, const size_t incY, typename Field::Element_ptr X, const size_t incX)`  
*fassign :  $x \leftarrow y$ .*
- `template<> void fassign (const Givaro::Modular< float > &F, const size_t N, const float *Y, const size_t incY, float *X, const size_t incX)`



- `template<> void fassign (const Givaro::ModularBalanced< float > &F, const size_t N, const float *Y, const size_t incY, float *X, const size_t incX)`
- `template<> void fassign (const Givaro::ZRing< float > &F, const size_t N, const float *Y, const size_t incY, float *X, const size_t incX)`
- `template<> void fassign (const Givaro::Modular< double > &F, const size_t N, const double *Y, const size_t incY, double *X, const size_t incX)`
- `template<> void fassign (const Givaro::ModularBalanced< double > &F, const size_t N, const double *Y, const size_t incY, double *X, const size_t incX)`
- `template<> void fassign (const Givaro::ZRing< double > &F, const size_t N, const double *Y, const size_t incY, double *X, const size_t incX)`
- `template<class Field >`  
`void fassign (const Field &F, const size_t m, const size_t n, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr A, const size_t lda)`  

$$fassign : A \leftarrow B.$$
- `template<class Field >`  
`void faxpy (const Field &F, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr X, const size_t incX, typename Field::Element_ptr Y, const size_t incY)`  

$$faxpy : y \leftarrow \alpha \cdot x + y.$$
- `template<> void faxpy (const Givaro::DoubleDomain &, const size_t N, const Givaro::DoubleDomain::Element a, Givaro::DoubleDomain::ConstElement_ptr x, const size_t incx, Givaro::DoubleDomain::Element_ptr y, const size_t incy)`
- `template<> void faxpy (const Givaro::FloatDomain &, const size_t N, const Givaro::FloatDomain::Element a, Givaro::FloatDomain::ConstElement_ptr x, const size_t incx, Givaro::FloatDomain::Element_ptr y, const size_t incy)`
- `template<class Field >`  
`void faxpy (const Field &F, const size_t m, const size_t n, const typename Field::Element alpha, typename Field::ConstElement_ptr X, const size_t idx, typename Field::Element_ptr Y, const size_t ldy)`  

$$faxpy : y \leftarrow \alpha \cdot x + y.$$
- `template<class Field >`  
`Field::Element fdot (const Field &F, const size_t N, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, ModeCategories::DefaultTag &MT)`
- `template<class Field >`  
`Field::Element fdot (const Field &F, const size_t N, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, ModeCategories::DelayedTag &MT)`
- `template<> Givaro::DoubleDomain::Element fdot (const Givaro::DoubleDomain &, const size_t N, Givaro::DoubleDomain::ConstElement_ptr x, const size_t incx, Givaro::DoubleDomain::ConstElement_ptr y, const size_t incy, ModeCategories::DefaultTag &MT)`
- `template<> Givaro::FloatDomain::Element fdot (const Givaro::FloatDomain &, const size_t N, Givaro::FloatDomain::ConstElement_ptr x, const size_t incx, Givaro::FloatDomain::ConstElement_ptr y, const size_t incy, ModeCategories::DefaultTag &MT)`
- `template<class Field, class T >`  
`Field::Element fdot (const Field &F, const size_t N, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, ModeCategories::ConvertTo< T > &MT)`
- `template<class Field >`  
`Field::Element fdot (const Field &F, const size_t N, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, ModeCategories::DefaultBoundedTag &dbt)`
- `template<class Field >`  
`Field::Element fdot (const Field &F, const size_t N, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, const ParSeqHelper::Sequential seq)`
- `template<class Field >`  
`Field::Element fdot (const Field &F, const size_t N, typename Field::ConstElement_ptr X, const size_t incX, typename Field::ConstElement_ptr Y, const size_t incY)`  

$$fdot: dot\ product\ x^T y.$$
- `template<class Field >`  
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename`

- `Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, ModeCategories::ConvertTo< ElementCategories::MachineFloatTag >, ParSeqHelper::Sequential > &H)`
- `template<typename Field >`  
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const ParSeqHelper::Sequential seq)`
  - `template<typename Field, class Cut, class Param >`  
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const ParSeqHelper::Parallel< Cut, Param > par)`
  - `template<typename Field >`  
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc)`  
*fgemm: Field GENERAL Matrix Multiply.*
  - `template<typename Field, class ModeT, class ParSeq >`  
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Auto, ModeT, ParSeq > &H)`
  - `template<class Field >`  
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, ModeCategories::DelayedTag, ParSeqHelper::Sequential > &H)`
  - `template<class Field >`  
`Field::Element_ptr fsquare (const Field &F, const FFLAS_TRANSPOSE ta, const size_t n, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc)`  
*fsquare: Squares a matrix.*
  - `template<> double * fsquare (const Givaro::ModularBalanced< double > &F, const FFLAS_TRANSPOSE ta, const size_t n, const double alpha, const double *A, const size_t lda, const double beta, double *C, const size_t ldc)`
  - `template<> float * fsquare (const Givaro::ModularBalanced< float > &F, const FFLAS_TRANSPOSE ta, const size_t n, const float alpha, const float *A, const size_t lda, const float beta, float *C, const size_t ldc)`
  - `template<> double * fsquare (const Givaro::Modular< double > &F, const FFLAS_TRANSPOSE ta, const size_t n, const double alpha, const double *A, const size_t lda, const double beta, double *C, const size_t ldc)`
  - `template<> float * fsquare (const Givaro::Modular< float > &F, const FFLAS_TRANSPOSE ta, const size_t n, const float alpha, const float *A, const size_t lda, const float beta, float *C, const size_t ldc)`
  - `template<typename RNS, typename ParSeqTrait >`  
`FFPACK::RNSInteger< RNS >::Element_ptr fgemm (const FFPACK::RNSInteger< RNS > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename FFPACK::RNSInteger< RNS >::Element alpha, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Ad, const size_t lda, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Bd, const size_t ldb, const typename FFPACK::RNSInteger< RNS >::Element beta, typename FFPACK::RNSInteger< RNS >::Element_ptr Cd, const size_t ldc, MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag, ParSeqHelper::Compose< ParSeqHelper::Sequential, ParSeqTrait > > &H)`

- `template<typename RNS >`  
`FFPACK::RNSInteger< RNS >::Element_ptr fgemm (const FFPACK::RNSInteger< RNS > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename FFPACK::RNSInteger< RNS >::Element alpha, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Ad, const size_t lda, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Bd, const size_t ldb, const typename FFPACK::RNSInteger< RNS >::Element beta, typename FFPACK::RNSInteger< RNS >::Element_ptr Cd, const size_t ldc, MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag, ParSeqHelper::Sequential > &H)`
- `template<typename RNS, typename ParSeqTrait >`  
`FFPACK::RNSInteger< RNS >::Element_ptr fgemm (const FFPACK::RNSInteger< RNS > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename FFPACK::RNSInteger< RNS >::Element alpha, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Ad, const size_t lda, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Bd, const size_t ldb, const typename FFPACK::RNSInteger< RNS >::Element beta, typename FFPACK::RNSInteger< RNS >::Element_ptr Cd, const size_t ldc, MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag, ParSeqHelper::Compose< ParSeqHelper::Parallel< CuttingStrategy::RNSModulus, StrategyParameter::Threads >, ParSeqTrait > > &H)`
- `template<typename RNS, typename Cut, typename Param >`  
`FFPACK::RNSInteger< RNS >::Element_ptr fgemm (const FFPACK::RNSInteger< RNS > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename FFPACK::RNSInteger< RNS >::Element alpha, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Ad, const size_t lda, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Bd, const size_t ldb, const typename FFPACK::RNSInteger< RNS >::Element beta, typename FFPACK::RNSInteger< RNS >::Element_ptr Cd, const size_t ldc, MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag, ParSeqHelper::Parallel< Cut, Param > > &H)`
- `template<class ParSeq >`  
`Givaro::Integer * fgemm (const Givaro::ZRing< Givaro::Integer > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const Givaro::Integer alpha, const Givaro::Integer *A, const size_t lda, const Givaro::Integer *B, const size_t ldb, Givaro::Integer beta, Givaro::Integer *C, const size_t ldc, MMHelper< Givaro::ZRing< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq > &H)`
- `template<typename RNS, class ModeT >`  
`RNS::Element_ptr fgemm (const FFPACK::RNSInteger< RNS > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename RNS::Element alpha, typename RNS::ConstElement_ptr Ad, const size_t lda, typename RNS::ConstElement_ptr Bd, const size_t ldb, const typename RNS::Element beta, typename RNS::Element_ptr Cd, const size_t ldc, MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Winograd, ModeT, ParSeqHelper::Sequential > &H)`
- `template<typename RNS >`  
`RNS::Element_ptr fgemm (const FFPACK::RNSIntegerMod< RNS > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename RNS::Element alpha, typename RNS::ConstElement_ptr Ad, const size_t lda, typename RNS::ConstElement_ptr Bd, const size_t ldb, const typename RNS::Element beta, typename RNS::Element_ptr Cd, const size_t ldc, MMHelper< FFPACK::RNSIntegerMod< RNS >, MMHelperAlgo::Winograd > &H)`
- `Givaro::Integer * fgemm (const Givaro::Modular< Givaro::Integer > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const Givaro::Integer alpha, const Givaro::Integer *A, const size_t lda, const Givaro::Integer *B, const size_t ldb, const Givaro::Integer beta, Givaro::Integer *C, const size_t ldc, MMHelper< Givaro::Modular< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::RNSElementTag > > &H)`
- `template<class ParSeq >`  
`Givaro::Integer * fgemm (const Givaro::Modular< Givaro::Integer > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const Givaro::Integer alpha, const Givaro::Integer *A, const size_t lda, const Givaro::Integer *B, const size_t ldb, const Givaro::Integer beta, Givaro::Integer *C, const size_t ldc, MMHelper< Givaro::Modular< Givaro::Integer >, MMHelperAlgo::Auto, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq > &H)`
- `template<size_t K1, size_t K2, class ParSeq >`  
`Reclnt::ruint< K1 > * fgemm (const Givaro::Modular< Reclnt::ruint< K1 >, Reclnt::ruint< K2 > > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n,`

- const size\_t k, const [RecInt::ruint](#)< K1 > alpha, const [RecInt::ruint](#)< K1 > \*A, const size\_t lda, const [RecInt::ruint](#)< K1 > \*B, const size\_t ldb, [RecInt::ruint](#)< K1 > beta, [RecInt::ruint](#)< K1 > \*C, const size\_t ldc, [MMHelper](#)< [Givaro::Modular](#)< [RecInt::ruint](#)< K1 >, [RecInt::ruint](#)< K2 > >, [MMHelperAlgo::Classic](#), [ModeCategories::ConvertTo](#)< [ElementCategories::RNSElementTag](#) >, [ParSeq](#) > &H)
- `template<class Field , class ModeT >`  
[Field::Element\\_ptr](#) [fgemm](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t m, const size\_t n, const size\_t k, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), ModeT > &H)
  - `template<class Field , class ModeT , class Cut , class Param >`  
[Field::Element\\_ptr](#) [fgemm](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t m, const size\_t n, const size\_t k, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, [MMHelper](#)< [Field](#), [MMHelperAlgo::WinogradPar](#), ModeT, [ParSeqHelper::Parallel](#)< Cut, Param > > &H)
  - `template<class Field >`  
[Field::Element\\_ptr](#) [fgemv](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) Y, const size\_t incY, [MMHelper](#)< [Field](#), [MMHelperAlgo::Classic](#), [ModeCategories::ConvertTo](#)< [ElementCategories::MachineFloatTag](#) > > &H)
  - `template<class Field >`  
[Field::Element\\_ptr](#) [fgemv](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) Y, const size\_t incY, [MMHelper](#)< [Field](#), [MMHelperAlgo::Classic](#), [ModeCategories::DelayedTag](#) > &H)
  - `template<class Field >`  
[Field::Element\\_ptr](#) [fgemv](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) Y, const size\_t incY, [MMHelper](#)< [Field](#), [MMHelperAlgo::Classic](#), [ModeCategories::DefaultTag](#) > &H)
  - `template<class Field >`  
[Field::Element\\_ptr](#) [fgemv](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) Y, const size\_t incY, [MMHelper](#)< [Field](#), [MMHelperAlgo::Classic](#), [ModeCategories::LazyTag](#) > &H)
  - `template<class Field >`  
[Field::Element\\_ptr](#) [fgemv](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) TransA, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) Y, const size\_t incY)  
*finite prime Field GEneral Matrix Vector multiplication.*
  - `Givaro::ZRing< int64_t >::Element_ptr` [fgemv](#) (const [Givaro::ZRing](#)< int64\_t > &F, const [FFLAS\\_TRANSPOSE](#) ta, const size\_t M, const size\_t N, const int64\_t alpha, const int64\_t \*A, const size\_t lda, const int64\_t \*X, const size\_t incX, const int64\_t beta, const int64\_t \*Y, const size\_t incY, [MMHelper](#)< [Givaro::ZRing](#)< int64\_t >, [MMHelperAlgo::Classic](#), [ModeCategories::DefaultTag](#) > &H)
  - `Givaro::DoubleDomain::Element_ptr` [fgemv](#) (const [Givaro::DoubleDomain](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const size\_t M, const size\_t N, const [Givaro::DoubleDomain::Element](#) alpha, const [Givaro::DoubleDomain::ConstElement\\_ptr](#) A, const size\_t lda, const [Givaro::DoubleDomain::ConstElement\\_ptr](#) X, const size\_t incX, const [Givaro::DoubleDomain::Element](#) beta, [Givaro::DoubleDomain::Element\\_ptr](#) Y, const size\_t incY, [MMHelper](#)< [Givaro::DoubleDomain](#), [MMHelperAlgo::Classic](#), [ModeCategories::DefaultTag](#) > &H)
  - `template<class Field >`  
[Field::Element\\_ptr](#) [fgemv](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) Y, const size\_t incY, [MMHelper](#)< [Field](#), [MMHelperAlgo::Classic](#), [ModeCategories::DefaultTag](#) > &H)

- `_t N, const typename Field::Element alpha, const typename Field::ConstElement_ptr A, const size_t lda, const typename Field::ConstElement_ptr X, const size_t incX, const typename Field::Element beta, typename Field::Element_ptr Y, const size_t incY, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DefaultBoundedTag > &H)`
- `Givaro::FloatDomain::Element_ptr fgemv (const Givaro::FloatDomain &F, const FFLAS_TRANSPOSE ta, const size_t M, const size_t N, const Givaro::FloatDomain::Element alpha, const Givaro::FloatDomain::ConstElement_ptr A, const size_t lda, const Givaro::FloatDomain::ConstElement_ptr X, const size_t incX, const Givaro::FloatDomain::Element beta, Givaro::FloatDomain::Element_ptr Y, const size_t incY, MMHelper< Givaro::FloatDomain, MMHelperAlgo::Classic, ModeCategories::DefaultTag > &H)`
- `template<class Field, class Cut, class Param >  
Field::Element_ptr fgemv (const Field &F, const FFLAS_TRANSPOSE ta, const size_t m, const size_t n, const typename Field::Element alpha, const typename Field::ConstElement_ptr A, const size_t lda, const typename Field::ConstElement_ptr X, const size_t incX, const typename Field::Element beta, typename Field::Element_ptr Y, const size_t incY, ParSeqHelper::Parallel< Cut, Param > &parH)`
- `template<class Field >  
Field::Element_ptr fgemv (const Field &F, const FFLAS_TRANSPOSE ta, const size_t m, const size_t n, const typename Field::Element alpha, const typename Field::ConstElement_ptr A, const size_t lda, const typename Field::ConstElement_ptr X, const size_t incX, const typename Field::Element beta, typename Field::Element_ptr Y, const size_t incY, ParSeqHelper::Sequential &seqH)`
- `FFPACK::rns_double::Element_ptr fgemv (const FFPACK::RNSInteger< FFPACK::rns_double > &F, const FFLAS_TRANSPOSE ta, const size_t M, const size_t N, const FFPACK::rns_double::Element alpha, FFPACK::rns_double::ConstElement_ptr A, const size_t lda, FFPACK::rns_double::ConstElement_ptr X, const size_t incX, const FFPACK::rns_double::Element beta, FFPACK::rns_double::Element_ptr Y, const size_t incY, MMHelper< FFPACK::RNSInteger< FFPACK::rns_double >, MMHelperAlgo::Classic, ModeCategories::DefaultTag > &H)`
- `FFPACK::rns_double::Element_ptr fgemv (const FFPACK::RNSIntegerMod< FFPACK::rns_double > &F, const FFLAS_TRANSPOSE ta, const size_t M, const size_t N, const FFPACK::rns_double::Element alpha, FFPACK::rns_double::ConstElement_ptr A, const size_t lda, FFPACK::rns_double::ConstElement_ptr X, const size_t incX, const FFPACK::rns_double::Element beta, FFPACK::rns_double::Element_ptr Y, const size_t incY, MMHelper< FFPACK::RNSIntegerMod< FFPACK::rns_double >, MMHelperAlgo::Classic, ModeCategories::DefaultTag > &H)`
- `Givaro::Integer * fgemv (const Givaro::ZRing< Givaro::Integer > &F, const FFLAS_TRANSPOSE ta, const size_t m, const size_t n, const Givaro::Integer alpha, Givaro::Integer *A, const size_t lda, Givaro::Integer *X, const size_t ldx, Givaro::Integer beta, Givaro::Integer *Y, const size_t ldy, MMHelper< Givaro::ZRing< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::RNSElementTag > > &H)`
- `Givaro::Integer * fgemv (const Givaro::Modular< Givaro::Integer > &F, const FFLAS_TRANSPOSE ta, const size_t m, const size_t n, const Givaro::Integer alpha, Givaro::Integer *A, const size_t lda, Givaro::Integer *X, const size_t ldx, Givaro::Integer beta, Givaro::Integer *Y, const size_t ldy, MMHelper< Givaro::Modular< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::RNSElementTag > > &H)`
- `template<size_t K1, size_t K2, class ParSeq >  
RecInt::ruint< K1 > * fgemv (const Givaro::Modular< RecInt::ruint< K1 >, RecInt::ruint< K2 > > &F, const FFLAS_TRANSPOSE ta, const size_t m, const size_t n, const RecInt::ruint< K1 > alpha, const RecInt::ruint< K1 > *A, const size_t lda, const RecInt::ruint< K1 > *X, const size_t incx, RecInt::ruint< K1 > beta, RecInt::ruint< K1 > *Y, const size_t incy, MMHelper< Givaro::Modular< RecInt::ruint< K1 >, RecInt::ruint< K2 > >, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq > &H)`
- `template<class Field >  
void fger (const Field &F, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, typename Field::Element_ptr A, const size_t lda)  
fger: rank one update of a general matrix`
- `template<class Field >  
void fger (const Field &F, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, typename Field::Element_ptr A, const size_t lda, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::MachineFloatTag > > &H)`



- `template<class Field, class AnyTag >`  
`void fger (const Field &F, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, typename Field::Element_ptr A, const size_t lda, MMHelper< Field, MMHelperAlgo::Classic, AnyTag > &H)`
- `void fger (const Givaro::DoubleDomain &F, const size_t M, const size_t N, const Givaro::DoubleDomain::Element alpha, const Givaro::DoubleDomain::ConstElement_ptr x, const size_t incx, const Givaro::DoubleDomain::ConstElement_ptr y, const size_t incy, Givaro::DoubleDomain::Element_ptr A, const size_t lda, MMHelper< Givaro::DoubleDomain, MMHelperAlgo::Classic, ModeCategories::DefaultTag > &H)`
- `template<class Field >`  
`void fger (const Field &F, const size_t M, const size_t N, const typename Field::Element alpha, const typename Field::ConstElement_ptr x, const size_t incx, const typename Field::ConstElement_ptr y, const size_t incy, typename Field::Element_ptr A, const size_t lda, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DefaultBoundedTag > &H)`
- `void fger (const Givaro::FloatDomain &F, const size_t M, const size_t N, const Givaro::FloatDomain::Element alpha, const Givaro::FloatDomain::ConstElement_ptr x, const size_t incx, const Givaro::FloatDomain::ConstElement_ptr y, const size_t incy, Givaro::FloatDomain::Element_ptr A, const size_t lda, MMHelper< Givaro::FloatDomain, MMHelperAlgo::Classic, ModeCategories::DefaultTag > &H)`
- `template<class Field >`  
`void fger (const Field &F, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, typename Field::Element_ptr A, const size_t lda, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::LazyTag > &H)`
- `template<class Field >`  
`void fger (const Field &F, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, typename Field::Element_ptr A, const size_t lda, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DelayedTag > &H)`
- `void fger (const Givaro::Modular< Givaro::Integer > &F, const size_t M, const size_t N, const typename Givaro::Integer alpha, typename Givaro::Integer *x, const size_t incx, typename Givaro::Integer *y, const size_t incy, typename Givaro::Integer *A, const size_t lda, MMHelper< Givaro::Modular< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::RNSElementTag > > &H)`
- `template<typename RNS >`  
`void fger (const FFPACK::RNSInteger< RNS > &F, const size_t M, const size_t N, const typename FFPACK::RNSInteger< RNS >::Element alpha, typename FFPACK::RNSInteger< RNS >::Element_ptr x, const size_t incx, typename FFPACK::RNSInteger< RNS >::Element_ptr y, const size_t incy, typename FFPACK::RNSInteger< RNS >::Element_ptr A, const size_t lda, MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag > &H)`
- `template<typename RNS >`  
`void fger (const FFPACK::RNSIntegerMod< RNS > &F, const size_t M, const size_t N, const typename FFPACK::RNSIntegerMod< RNS >::Element alpha, typename FFPACK::RNSIntegerMod< RNS >::Element_ptr x, const size_t incx, typename FFPACK::RNSIntegerMod< RNS >::Element_ptr y, const size_t incy, typename FFPACK::RNSIntegerMod< RNS >::Element_ptr A, const size_t lda, MMHelper< FFPACK::RNSIntegerMod< RNS >, MMHelperAlgo::Classic > &H)`
- `template<class Field >`  
`void freduce (const Field &F, const size_t n, typename Field::ConstElement_ptr Y, const size_t incY, typename Field::Element_ptr X, const size_t incX)`  

$$\text{freduce } x \leftarrow y \bmod F.$$
- `template<class Field >`  
`void freduce (const Field &F, const size_t n, typename Field::Element_ptr X, const size_t incX)`  

$$\text{freduce } x \leftarrow x \bmod F.$$
- `template<class Field >`  
`void freduce_constoverride (const Field &F, const size_t m, typename Field::ConstElement_ptr A, const size_t incX)`
- `template<class Field, class ConstOtherElement_ptr >`  
`void finit (const Field &F, const size_t n, ConstOtherElement_ptr Y, const size_t incY, typename Field::Element_ptr X, const size_t incX)`

- template<class [Field](#) >  
void [finit](#) (const [Field](#) &F, const size\_t n, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
*finit Initializes  $X$  in  $F^{\$}$ .*
- template<class [Field](#) >  
void [freduce](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*freduce  $A \leftarrow A \bmod F$ .*
- template<class [Field](#) >  
void [freduce](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) uplo, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*freduce for square symmetric matrices*
- template<class [Field](#) >  
void [pfreduce](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t numths)
- template<class [Field](#) >  
void [freduce](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*freduce  $A \leftarrow B \bmod F$ .*
- template<class [Field](#) >  
void [freduce\\_constoverride](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda)
- template<class [Field](#) , class OtherElement\_ptr >  
void [finit](#) (const [Field](#) &F, const size\_t m, const size\_t n, const OtherElement\_ptr B, const size\_t ldb, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*finit  $A \leftarrow B \bmod F$ .*
- template<class [Field](#) >  
void [finit](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)
- template<> void [freduce](#) (const [FFPACK::RNSIntegerMod](#)< [FFPACK::rns\\_double](#) > &F, const size\_t n, [FFPACK::RNSIntegerMod](#)< [FFPACK::rns\\_double](#) >::Element\_ptr A, size\_t inc)
- template<> void [freduce](#) (const [FFPACK::RNSIntegerMod](#)< [FFPACK::rns\\_double](#) > &F, const size\_t m, const size\_t n, [FFPACK::rns\\_double::Element\\_ptr](#) A, size\_t lda)
- template<class [Field](#) >  
bool [freivalds](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t m, const size\_t n, const size\_t k, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::ConstElement\\_ptr](#) C, const size\_t ldc)  
*freivalds: **Freivalds** **GE**neral **M**atrix **M**ultiply **R**andom **C**heck.*
- template<class [Field](#) >  
void [fscal](#) (const [Field](#) &F, const size\_t n, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
*fscal  $x \leftarrow \alpha \cdot x$ .*
- template<class [Field](#) >  
void [fscal](#) (const [Field](#) &F, const size\_t n, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, typename [Field::Element\\_ptr](#) Y, const size\_t incY)  
*fscal  $y \leftarrow \alpha \cdot x$ .*
- template<> void [fscal](#) (const Givaro::DoubleDomain &, const size\_t N, const Givaro::DoubleDomain::Element a, Givaro::DoubleDomain::ConstElement\_ptr x, const size\_t incx, Givaro::DoubleDomain::Element\_ptr y, const size\_t incy)
- template<> void [fscal](#) (const Givaro::FloatDomain &, const size\_t N, const Givaro::FloatDomain::Element a, Givaro::FloatDomain::ConstElement\_ptr x, const size\_t incx, Givaro::FloatDomain::Element\_ptr y, const size\_t incy)
- template<> void [fscal](#) (const Givaro::DoubleDomain &, const size\_t N, const Givaro::DoubleDomain::Element a, Givaro::DoubleDomain::Element\_ptr y, const size\_t incy)
- template<> void [fscal](#) (const Givaro::FloatDomain &, const size\_t N, const Givaro::FloatDomain::Element a, Givaro::FloatDomain::Element\_ptr y, const size\_t incy)

- `template<class Field >`  
`void fscaln (const Field &F, const size_t m, const size_t n, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda)`  

$$fscaln\ A \leftarrow a \cdot A.$$
- `template<class Field >`  
`void fscal (const Field &F, const size_t m, const size_t n, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb)`  

$$fscal\ B \leftarrow a \cdot A.$$
- `template<> void fscaln (const FFPACK::RNSInteger< FFPACK::rns_double > &F, const size_t n, const FFPACK::rns_double::Element alpha, FFPACK::rns_double::Element_ptr A, const size_t inc)`
- `template<> void fscal (const FFPACK::RNSInteger< FFPACK::rns_double > &F, const size_t n, const FFPACK::rns_double::Element alpha, FFPACK::rns_double::ConstElement_ptr A, const size_t Ainc, FFPACK::rns_double::Element_ptr B, const size_t Binc)`
- `template<> void fscaln (const FFPACK::RNSInteger< FFPACK::rns_double > &F, const size_t m, const size_t n, const FFPACK::rns_double::Element alpha, FFPACK::rns_double::Element_ptr A, const size_t lda)`
- `template<> void fscal (const FFPACK::RNSInteger< FFPACK::rns_double > &F, const size_t m, const size_t n, const FFPACK::rns_double::Element alpha, FFPACK::rns_double::ConstElement_ptr A, const size_t Ainc, FFPACK::rns_double::Element_ptr B, const size_t Binc)`
- `template<> void fscaln (const FFPACK::RNSIntegerMod< FFPACK::rns_double > &F, const size_t n, const typename FFPACK::RNSIntegerMod< FFPACK::rns_double >::Element alpha, typename FFPACK::RNSIntegerMod< FFPACK::rns_double >::Element_ptr A, const size_t inc)`
- `template<> void fscal (const FFPACK::RNSIntegerMod< FFPACK::rns_double > &F, const size_t n, const FFPACK::rns_double::Element alpha, FFPACK::rns_double::ConstElement_ptr A, const size_t Ainc, FFPACK::rns_double::Element_ptr B, const size_t Binc)`
- `template<> void fscaln (const FFPACK::RNSIntegerMod< FFPACK::rns_double > &F, const size_t m, const size_t n, const FFPACK::rns_double::Element alpha, FFPACK::rns_double::Element_ptr A, const size_t lda)`
- `template<> void fscal (const FFPACK::RNSIntegerMod< FFPACK::rns_double > &F, const size_t m, const size_t n, const FFPACK::rns_double::Element alpha, FFPACK::rns_double::ConstElement_ptr A, const size_t Ainc, FFPACK::rns_double::Element_ptr B, const size_t Binc)`
- `template<class Field >`  
`Field::Element_ptr fsyr2k (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc)`  

$$fsyr2k: \text{Symmetric Rank } 2K \text{ update}$$
- `template<class Field >`  
`Field::Element_ptr fsyrk (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc)`  

$$fsyrk: \text{Symmetric Rank } K \text{ update}$$
- `template<class Field >`  
`Field::Element_ptr fsyrk (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const size_t N, const size_t K, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const ParSeqHelper::Sequential seq)`
- `template<class Field >`  
`Field::Element_ptr fsyrk (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const size_t N, const size_t K, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DefaultTag > &H)`
- `template<class Field >`  
`Field::Element_ptr fsyrk (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const size_t N, const size_t K, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::MachineFloatTag >, ParSeqHelper::Sequential > &H)`



- `template<class Field >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` A, const `size_t` lda, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, `MMHelper`< `Field`, `MMHelperAlgo::Classic`, `ModeCategories::DelayedTag` > &H)
- `template<class Field >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` A, const `size_t` lda, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, `MMHelper`< `Field`, `MMHelperAlgo::Classic`, `ModeCategories::LazyTag` > &H)
- `template<class Field , typename Mode >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` A, const `size_t` lda, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, `MMHelper`< `Field`, `MMHelperAlgo::DivideAndConquer`, `Mode` > &H)
- `template<class Field >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` A, const `size_t` lda, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, `MMHelper`< `Field`, `MMHelperAlgo::Classic`, `ModeCategories::DefaultBoundedTag` > &H)
- `Givaro::FloatDomain::Element_ptr fsyrk` (const `Givaro::FloatDomain` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const `Givaro::FloatDomain::Element` alpha, `Givaro::FloatDomain::ConstElement_ptr` A, const `size_t` lda, const `Givaro::FloatDomain::Element` beta, `Givaro::FloatDomain::Element_ptr` C, const `size_t` ldc, `MMHelper`< `Givaro::FloatDomain`, `MMHelperAlgo::Classic`, `ModeCategories::DefaultTag` > &H)
- `Givaro::DoubleDomain::Element_ptr fsyrk` (const `Givaro::DoubleDomain` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const `Givaro::DoubleDomain::Element` alpha, `Givaro::DoubleDomain::ConstElement_ptr` A, const `size_t` lda, const `Givaro::DoubleDomain::Element` beta, `Givaro::DoubleDomain::Element_ptr` C, const `size_t` ldc, `MMHelper`< `Givaro::DoubleDomain`, `MMHelperAlgo::Classic`, `ModeCategories::DefaultTag` > &H)
- `template<class Field >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` n, const `size_t` k, const typename `Field::Element` alpha, typename `Field::Element_ptr` A, const `size_t` lda, typename `Field::ConstElement_ptr` D, const `size_t` incD, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, const `size_t` threshold=`__FFLASFFPACK_FSYRK_THRESHOLD`)  
*fsyrk: Symmetric Rank K update with diagonal scaling*
- `template<class Field >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const typename `Field::Element` alpha, typename `Field::Element_ptr` A, const `size_t` lda, typename `Field::ConstElement_ptr` D, const `size_t` incD, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, const `ParSeqHelper::Sequential` seq, const `size_t` threshold)
- `template<class Field , class Cut , class Param >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const typename `Field::Element` alpha, typename `Field::Element_ptr` A, const `size_t` lda, typename `Field::ConstElement_ptr` D, const `size_t` incD, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, const `ParSeqHelper::Parallel`< `Cut`, `Param` > par, const `size_t` threshold)
- `template<class Field >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` n, const `size_t` k, const typename `Field::Element` alpha, typename `Field::Element_ptr` A, const `size_t` lda, typename `Field::ConstElement_ptr` D, const `size_t` incD, const `std::vector< bool >` &twoBlock, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, const `size_t` threshold=`__FFLASFFPACK_FSYRK_THRESHOLD`)  
*fsyrk: Symmetric Rank K update with diagonal scaling*
- `template<class Field , class FieldTrait >`  
void `computeS1S2` (const `Field` &F, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const

- typename [Field::Element](#) x, const typename [Field::Element](#) y, typename [Field::Element\\_ptr](#) A, const size\_t Ida, typename [Field::Element\\_ptr](#) S, const size\_t Ids, typename [Field::Element\\_ptr](#) T, const size\_t Idt, [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), [FieldTrait](#) > &WH)
- template<class [Field](#) >  
[Field::Element\\_ptr](#) fsyrk (const [Field](#) &F, const [FFLAS\\_UPLO](#) UpLo, const [FFLAS\\_TRANSPOSE](#) trans, const size\_t N, const size\_t K, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t Ida, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t Idc, [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), [ModeCategories::DelayedTag](#), [ParSeqHelper::Sequential](#) > &H)
  - template<class [Field](#) , class Mode >  
[Field::Element\\_ptr](#) fsyrk (const [Field](#) &F, const [FFLAS\\_UPLO](#) UpLo, const [FFLAS\\_TRANSPOSE](#) trans, const size\_t N, const size\_t K, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t Ida, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t Idc, [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), Mode > &H)
  - template<class [Field](#) , class [FieldTrait](#) >  
[Field::Element\\_ptr](#) fsyrk\_strassen (const [Field](#) &F, const [FFLAS\\_UPLO](#) uplo, const [FFLAS\\_TRANSPOSE](#) trans, const size\_t N, const size\_t K, const typename [Field::Element](#) y1, const typename [Field::Element](#) y2, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) A, const size\_t Ida, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t Idc, [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), [FieldTrait](#) > &WH)
  - template<class [Field](#) >  
void ftrmm (const [Field](#) &F, const [FFLAS\\_SIDE](#) Side, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t Ida, typename [Field::Element\\_ptr](#) B, const size\_t ldb)  
*ftrmm: **TRI**angular **M**atrix **M**ultiply.*
  - template<class [Field](#) >  
void ftrmm (const [Field](#) &F, const [FFLAS\\_SIDE](#) Side, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t Ida, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc)  
*ftrmm: **TRI**angular **M**atrix **M**ultiply with 3 operands Computes  $C \leftarrow \alpha \text{op}(A)B + \text{beta}C$  or  $C \leftarrow \alpha \text{Bop}(A) + \text{beta}C$ .*
  - template<class [Field](#) >  
void ftrsm (const [Field](#) &F, const [FFLAS\\_SIDE](#) Side, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) A, const size\_t Ida, typename [Field::Element\\_ptr](#) B, const size\_t ldb)
  - template<class [Field](#) >  
void ftrsm (const [Field](#) &F, const [FFLAS\\_SIDE](#) Side, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) A, const size\_t Ida, typename [Field::Element\\_ptr](#) B, const size\_t ldb, const [ParSeqHelper::Sequential](#) &PSH)
  - template<class [Field](#) , class Cut , class Param >  
void ftrsm (const [Field](#) &F, const [FFLAS\\_SIDE](#) Side, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) A, const size\_t Ida, typename [Field::Element\\_ptr](#) B, const size\_t ldb, const [ParSeqHelper::Parallel](#)< Cut, Param > &PSH)
  - template<class [Field](#) , class [ParSeqTrait](#) = [ParSeqHelper::Sequential](#)>  
void ftrsm (const [Field](#) &F, const [FFLAS\\_SIDE](#) Side, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) A, const size\_t Ida, typename [Field::Element\\_ptr](#) B, const size\_t ldb, [TRSMHelper](#)< [StructureHelper::Recursive](#), [ParSeqTrait](#) > &H)
  - void ftrsm (const [Givaro::Modular](#)< [Givaro::Integer](#) > &F, const [FFLAS\\_SIDE](#) Side, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, const [Givaro::Integer](#) alpha, const [Givaro::Integer](#) \*A, const size\_t Ida, [Givaro::Integer](#) \*B, const size\_t ldb)
  - void cblas\_imptrsm (const enum [FFLAS\\_ORDER](#) Order, const enum [FFLAS\\_SIDE](#) Side, const enum [FFLAS\\_UPLO](#) Uplo, const enum [FFLAS\\_TRANSPOSE](#) TransA, const enum [FFLAS\\_DIAG](#) Diag, const int M, const int N, const [FFPACK::rns\\_double\\_elt](#) alpha, [FFPACK::rns\\_double\\_elt\\_cstptr](#) A, const int Ida, [FFPACK::rns\\_double\\_elt\\_ptr](#) B, const int ldb)

- template<class [Field](#) >  
void [ftrsv](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) X, int incX)  
*ftrsv: TRIangular System solve with Vector Computes  $X \leftarrow \text{op}(A^{-1})X$*
- void [igemm](#) (const enum [FFLAS\\_ORDER](#) Order, const enum [FFLAS\\_TRANSPOSE](#) TransA, const enum [FFLAS\\_TRANSPOSE](#) TransB, const size\_t M, const size\_t N, const size\_t K, const int64\_t alpha, const int64\_t \*A, const size\_t lda, const int64\_t \*B, const size\_t ldb, const int64\_t beta, int64\_t \*C, const size\_t ldc)
- template<class [Field](#) , class OtherElement\_ptr >  
void [finit](#) (const [Field](#) &F, const size\_t n, const OtherElement\_ptr Y, const size\_t incY, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
*finit  $x \leftarrow y \bmod F$ .*
- template<class [Field](#) , class OtherElement\_ptr >  
void [fconvert](#) (const [Field](#) &F, const size\_t n, OtherElement\_ptr X, const size\_t incX, typename [Field::ConstElement\\_ptr](#) Y, const size\_t incY)  
*fconvert  $x \leftarrow y \bmod F$ .*
- template<class [Field](#) >  
void [fnegin](#) (const [Field](#) &F, const size\_t n, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
*fnegin  $x \leftarrow -x$ .*
- template<class [Field](#) >  
void [fneg](#) (const [Field](#) &F, const size\_t n, typename [Field::ConstElement\\_ptr](#) Y, const size\_t incY, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
*fneg  $x \leftarrow -y$ .*
- template<class [Field](#) >  
void [fzero](#) (const [Field](#) &F, const size\_t n, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
*fzero :  $A \leftarrow 0$ .*
- template<class [Field](#) , class Randlter >  
void [frand](#) (const [Field](#) &F, Randlter &G, const size\_t n, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
*frand :  $A \leftarrow \text{random}$ .*
- template<class [Field](#) >  
bool [fiszero](#) (const [Field](#) &F, const size\_t n, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX)  
*fiszero : test  $X = 0$ .*
- template<class [Field](#) >  
bool [fequal](#) (const [Field](#) &F, const size\_t n, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, typename [Field::ConstElement\\_ptr](#) Y, const size\_t incY)  
*fequal : test  $X = Y$ .*
- template<class [Field](#) >  
void [faxpby](#) (const [Field](#) &F, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) Y, const size\_t incY)  
*faxpby :  $y \leftarrow \alpha \cdot x + \beta \cdot y$ .*
- template<typename [Field](#) , class Cut , class Param >  
[Field::Element](#) [fdot](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, typename [Field::ConstElement\\_ptr](#) Y, const size\_t incY, const [ParSeqHelper::Parallel](#)< Cut, Param > par)
- template<class [Field](#) >  
void [fswap](#) (const [Field](#) &F, const size\_t N, typename [Field::Element\\_ptr](#) X, const size\_t incX, typename [Field::Element\\_ptr](#) Y, const size\_t incY)  
*fswap:  $X \leftrightarrow Y$ .*
- template<class [Field](#) >  
void [fzero](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*fzero :  $A \leftarrow 0$ .*
- template<class [Field](#) >  
void [fzero](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) shape, const [FFLAS\\_DIAG](#) diag, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)

- $fzero : A \leftarrow 0$  for a triangular matrix.
- template<class [Field](#) , class RandIter >  
 void [frand](#) (const [Field](#) &F, RandIter &G, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
 $frand : A \leftarrow random.$
- template<class [Field](#) >  
 bool [fequal](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb)  
 $fequal : test A = B.$
- template<class [Field](#) >  
 bool [fiszero](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda)  
 $fiszero : test A = 0.$
- template<class [Field](#) >  
 void [fidentity](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda, const typename [Field::Element](#) &d)  
 $creates a diagonal matrix$
- template<class [Field](#) >  
 void [fidentity](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
 $creates a diagonal matrix$
- template<class [Field](#) , class OtherElement\_ptr >  
 void [finit](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
 $finit$  Initializes  $A$  in  $F^S$ .
- template<class [Field](#) , class OtherElement\_ptr >  
 void [fconvert](#) (const [Field](#) &F, const size\_t m, const size\_t n, OtherElement\_ptr A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb)  
 $fconvert A \leftarrow B mod F.$
- template<class [Field](#) >  
 void [fnegin](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
 $fnegin A \leftarrow -A.$
- template<class [Field](#) >  
 void [fneg](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
 $fneg A \leftarrow -B.$
- template<class [Field](#) >  
 void [faxpby](#) (const [Field](#) &F, const size\_t m, const size\_t n, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) X, const size\_t ldx, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) Y, const size\_t ldy)  
 $faxpby : y \leftarrow \alpha \cdot x + \beta \cdot y.$
- template<class [Field](#) >  
 void [fmove](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) B, const size\_t ldb)  
 $fmove : A \leftarrow B \text{ and } B \leftarrow 0.$
- template<class [Field](#) >  
 size\_t [bitsize](#) (const [Field](#) &F, size\_t M, size\_t N, const typename [Field::ConstElement\\_ptr](#) A, size\_t lda)  
 $bitsize$ : Computes the largest bitsize of the matrix' coefficients.
- template<> size\_t [bitsize](#)< [Givaro::ZRing](#)< [Givaro::Integer](#) > > (const [Givaro::ZRing](#)< [Givaro::Integer](#) > &F, size\_t M, size\_t N, const [Givaro::Integer](#) \*A, size\_t lda)
- template<class [Field](#) >  
 void [ftsmv](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) X, int incX)  
 $ftsm$ : TRIangular Matrix Vector prodcut Computes  $X \leftarrow op(A)X$

- `template<class Field >`  
`void ftrsm (const Field &F, const FFLAS_SIDE Side, const FFLAS_UPLO Uplo, const FFLAS_TRANSPOSE TransA, const FFLAS_DIAG Diag, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb)`  
*ftrsm: **TR**iangular **S**ystem solve with **M**atrix.*
- `template<class Field , typename FieldTrait >`  
`Field::Element_ptr fsyrk_strassen (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const size_t N, const size_t K, const typename Field::Element y1, const typename Field::Element y2, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &H)`
- `template<typename Field >`  
`Field::Element_ptr pfgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, size_t numthreads=0)`
- `template<class Field >`  
`Field::Element * pfgemm_1D_rec (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename Field::Element_ptr A, const size_t lda, const typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element *C, const size_t ldc, size_t seuil)`
- `template<class Field >`  
`Field::Element * pfgemm_2D_rec (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename Field::Element_ptr A, const size_t lda, const typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element *C, const size_t ldc, size_t seuil)`
- `template<class Field >`  
`Field::Element * pfgemm_3D_rec (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename Field::Element_ptr A, const size_t lda, const typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, size_t seuil, size_t *x)`
- `template<class Field >`  
`Field::Element_ptr pfgemm_3D_rec2 (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename Field::Element_ptr A, const size_t lda, const typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, size_t seuil, size_t *x)`
- `template<class Field , class ModeTrait , class Strat , class Param >`  
`std::enable_if<!std::is_same< ModeTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >::value, typename Field::Element_ptr >::type>::type fgemm (const Field &F, const FFLAS::FFLAS_TRANSPOSE ta, const FFLAS::FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, ModeTrait, ParSeqHelper::Parallel< Strat, Param > > &H)`
- `template<class Field , class Cut , class Param >`  
`Field::Element_ptr ftrsm (const Field &F, const FFLAS::FFLAS_SIDE Side, const FFLAS::FFLAS_UPLO UpLo, const FFLAS::FFLAS_TRANSPOSE TA, const FFLAS::FFLAS_DIAG Diag, const size_t m, const size_t n, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, TRSMHelper< StructureHelper::Iterative, ParSeqHelper::Parallel< Cut, Param > > &H)`
- `template<class Field , class Cut , class Param >`  
`Field::Element_ptr ftrsm (const Field &F, const FFLAS::FFLAS_SIDE Side, const FFLAS::FFLAS_UPLO UpLo, const FFLAS::FFLAS_TRANSPOSE TA, const FFLAS::FFLAS_DIAG Diag, const size_t m, const size_t n, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, TRSMHelper< StructureHelper::Hybrid, ParSeqHelper::Parallel< Cut, Param > > &H)`

- `template<class Field , class SM >`  
`void fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, const typename Field::Element &beta, typename Field::Element_ptr y)`
- `template<class Field , class SM >`  
`void fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, const typename Field::Element &beta, typename Field::Element_ptr y, int ldy)`
- `template<class Field , class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::COO > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field , class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::COO_ZO > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::COO > &A)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::COO_ZO > &A)`
- `template<class Field , class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::CSR > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field , class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::CSR_ZO > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::CSR > &A)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::CSR_ZO > &A)`
- `template<class Field >`  
`std::ostream & sparse_print (std::ostream &os, const Sparse< Field, SparseMatrix_t::CSR > &A)`
- `template<class IndexT >`  
`void sparse_init (const Givaro::Modular< Givaro::Integer > &F, Sparse< Givaro::Modular< Givaro::Integer >, SparseMatrix_t::CSR > &A, const IndexT *row, const IndexT *col, Givaro::Integer *dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class IndexT >`  
`void sparse_init (const Givaro::ZRing< Givaro::Integer > &F, Sparse< Givaro::ZRing< Givaro::Integer >, SparseMatrix_t::CSR_ZO > &A, const IndexT *row, const IndexT *col, Givaro::Integer *dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class IndexT , size_t RECINT_SIZE>`  
`void sparse_init (const Givaro::ZRing< RecInt::rmint< RECINT_SIZE > > &F, Sparse< Givaro::ZRing< RecInt::rmint< RECINT_SIZE > >, SparseMatrix_t::CSR_ZO > &A, const IndexT *row, const IndexT *col, typename Givaro::ZRing< RecInt::rmint< RECINT_SIZE > >::Element_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class IndexT , size_t RECINT_SIZE>`  
`void sparse_init (const Givaro::ZRing< RecInt::rmint< RECINT_SIZE > > &F, Sparse< Givaro::ZRing< RecInt::rmint< RECINT_SIZE > >, SparseMatrix_t::CSR > &A, const IndexT *row, const IndexT *col, typename Givaro::ZRing< RecInt::rmint< RECINT_SIZE > >::Element_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::CSR_HYB > &A)`
- `template<class Field , class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::CSR_HYB > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field , class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::ELL > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field , class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::ELL_ZO > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`



- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::ELL > &A)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::ELL_ZO > &A)`
- `template<class Field , class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::ELL_simd > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field , class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::ELL_simd > &A)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A)`
- `template<class Field >`  
`void sparse_print (const Sparse< Field, SparseMatrix_t::ELL_simd > &A)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::HYB_ZO > &A)`
- `template<class Field , class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::HYB_ZO > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<typename _Field >`  
`std::ostream & operator<< (std::ostream &os, const Sparse< _Field, SparseMatrix_t::HYB_ZO > &A)`
- `template<class Field , bool sorted = true, bool read_integer = false>`  
`void readSmsFormat (const std::string &path, const Field &f, index_t *&row, index_t *&col, typename Field::Element_ptr &val, index_t &rowdim, index_t &coldim, uint64_t &nnz)`
- `template<class Field >`  
`void readSprFormat (const std::string &path, const Field &f, index_t *&row, index_t *&col, typename Field::Element_ptr &val, index_t &rowdim, index_t &coldim, uint64_t &nnz)`
- `template<class T >`  
`std::enable_if< std::is_integral< T >::value, int > getDataType ()`
- `template<class T >`  
`std::enable_if< std::is_floating_point< T >::value, int > getDataType ()`
- `template<class T >`  
`std::enable_if< std::is_same< T, mpz_t >::value, int > getDataType ()`
- `template<class T >`  
`int getDataType ()`
- `template<class Field >`  
`void readMachineType (const Field &F, typename Field::Element &modulo, typename Field::Element_ptr val, std::ifstream &file, const uint64_t dims, const mask_t data_type, const mask_t field_desc)`
- `template<class Field >`  
`void readDnsFormat (const std::string &path, const Field &F, index_t &rowdim, index_t &coldim, typename Field::Element_ptr &val)`
- `template<class Field >`  
`void writeDnsFormat (const std::string &path, const Field &F, const index_t &rowdim, const index_t &coldim, typename Field::Element_ptr A, index_t ldA)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::SELL > &A)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::SELL_ZO > &A)`
- `template<class Field >`  
`void sparse_print (const Sparse< Field, SparseMatrix_t::SELL > &A)`

- `template<class Field, class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::SELL > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz, uint64_t sigma=0)`
- `template<class Field, class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::SELL_ZO > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class It >`  
`double computeDeviation (It begin, It end)`
- `template<class Field >`  
`StatsMatrix getStat (const Field &F, const index_t *row, const index_t *col, typename Field::ConstElement_ptr val, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field, class enable = void>`  
`Field::Residu_t maxCardinality ()`
- `template<> uint64_t maxCardinality< Givaro::Modular< int64_t > > ()`
- `template<> uint32_t maxCardinality< Givaro::Modular< int32_t > > ()`
- `template<class Field >`  
`Field::Residu_t minCardinality ()`
- `template<> void fflas_delete (FFPACK::rns_double_elt_ptr A)`
- `template<> void fflas_delete (FFPACK::rns_double_elt_cstptr A)`
- `template<> FFPACK::rns_double_elt_ptr fflas_new (const FFPACK::RNSIntegerMod< FFPACK::rns_double > &F, const size_t m, const Alignment align)`
- `template<> FFPACK::rns_double_elt_ptr fflas_new (const FFPACK::RNSIntegerMod< FFPACK::rns_double > &F, const size_t m, const size_t n, const Alignment align)`
- `template<typename RNS >`  
`void finit_rns (const FFPACK::RNSIntegerMod< RNS > &F, const size_t m, const size_t n, size_t k, const Givaro::Integer *B, const size_t ldb, typename RNS::Element_ptr A)`
- `template<typename RNS >`  
`void finit_trans_rns (const FFPACK::RNSIntegerMod< RNS > &F, const size_t m, const size_t n, size_t k, const Givaro::Integer *B, const size_t ldb, typename RNS::Element_ptr A)`
- `template<typename RNS >`  
`void fconvert_rns (const FFPACK::RNSIntegerMod< RNS > &F, const size_t m, const size_t n, Givaro::Integer alpha, Givaro::Integer *B, const size_t ldb, typename RNS::ConstElement_ptr A)`
- `template<typename RNS >`  
`void fconvert_trans_rns (const FFPACK::RNSIntegerMod< RNS > &F, const size_t m, const size_t n, Givaro::Integer alpha, Givaro::Integer *B, const size_t ldb, typename RNS::ConstElement_ptr A)`
- `template<> FFPACK::rns_double_elt_ptr fflas_new (const FFPACK::RNSInteger< FFPACK::rns_double > &F, const size_t m, const Alignment align)`
- `template<> FFPACK::rns_double_elt_ptr fflas_new (const FFPACK::RNSInteger< FFPACK::rns_double > &F, const size_t m, const size_t n, const Alignment align)`
- `template<typename RNS >`  
`void finit_rns (const FFPACK::RNSInteger< RNS > &F, const size_t m, const size_t n, size_t k, const Givaro::Integer *B, const size_t ldb, typename FFPACK::RNSInteger< RNS >::Element_ptr A)`
- `template<typename RNS >`  
`void fconvert_rns (const FFPACK::RNSInteger< RNS > &F, const size_t m, const size_t n, Givaro::Integer alpha, Givaro::Integer *B, const size_t ldb, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr A)`
- `template INST_OR_DECL void freduce (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t n, const FFLAS_ELT *Y, const size_t incY, FFLAS_ELT *X, const size_t incX)`  

$$\text{freduce } x \leftarrow x \bmod F.$$
- `template INST_OR_DECL void freduce (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t n, const FFLAS_ELT *Y, const size_t incY, FFLAS_ELT *X, const size_t incX)`  

$$\text{freduce } x \leftarrow y \bmod F.$$
- `template INST_OR_DECL void finit (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t n, const FFLAS_ELT *Y, const size_t incY, FFLAS_ELT *X, const size_t incX)`  

$$\text{finit } x \leftarrow y \bmod F.$$



- template `INST_OR_DECL` void `fconvert` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` n, `FFLAS_ELT` \*X, const `size_t` incX, const `FFLAS_ELT` \*Y, const `size_t` incY)  
 $fconvert\ x \leftarrow y \bmod F.$
- template `INST_OR_DECL` void `fnegin` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` n, `FFLAS_ELT` \*X, const `size_t` incX)  
 $fnegin\ x \leftarrow -x.$
- template `INST_OR_DECL` void `fneg` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` n, const `FFLAS_ELT` \*Y, const `size_t` incY, `FFLAS_ELT` \*X, const `size_t` incX)  
 $fneg\ x \leftarrow -y.$
- template `INST_OR_DECL` void `fzero` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` n, `FFLAS_ELT` \*X, const `size_t` incX)  
 $fzero : A \leftarrow 0.$
- template `INST_OR_DECL` bool `fiszero` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` n, const `FFLAS_ELT` \*X, const `size_t` incX)  
 $fiszero : test\ X = 0.$
- template `INST_OR_DECL` bool `fequal` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` n, const `FFLAS_ELT` \*X, const `size_t` incX, const `FFLAS_ELT` \*Y, const `size_t` incY)  
 $fequal : test\ X = Y.$
- template `INST_OR_DECL` void `fassign` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` N, const `FFLAS_ELT` \*Y, const `size_t` incY, `FFLAS_ELT` \*X, const `size_t` incX)  
 $fassign : x \leftarrow y.$
- template `INST_OR_DECL` void `fscaln` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` n, const `FFLAS_ELT` alpha, `FFLAS_ELT` \*X, const `size_t` incX)  
 $fscaln\ x \leftarrow \alpha \cdot x.$
- template `INST_OR_DECL` void `fscal` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` n, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*X, const `size_t` incX, `FFLAS_ELT` \*Y, const `size_t` incY)  
 $fscal\ y \leftarrow \alpha \cdot x.$
- template `INST_OR_DECL` void `faxpy` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` N, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*X, const `size_t` incX, `FFLAS_ELT` \*Y, const `size_t` incY)  
 $faxpy : y \leftarrow \alpha \cdot x + y.$
- template `INST_OR_DECL` `FFLAS_ELT` `fdot` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` N, const `FFLAS_ELT` \*X, const `size_t` incX, const `FFLAS_ELT` \*Y, const `size_t` incY)  
 $faxpby : y \leftarrow \alpha \cdot x + \beta \cdot y.$
- template `INST_OR_DECL` void `fswap` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` N, `FFLAS_ELT` \*X, const `size_t` incX, `FFLAS_ELT` \*Y, const `size_t` incY)  
 $fswap : X \leftrightarrow Y.$
- template `INST_OR_DECL` void `fadd` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` N, const `FFLAS_ELT` \*A, const `size_t` inca, const `FFLAS_ELT` \*B, const `size_t` incb, `FFLAS_ELT` \*C, const `size_t` incc)
- template `INST_OR_DECL` void `fsub` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` N, const `FFLAS_ELT` \*A, const `size_t` inca, const `FFLAS_ELT` \*B, const `size_t` incb, `FFLAS_ELT` \*C, const `size_t` incc)
- template `INST_OR_DECL` void `faddn` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` N, const `FFLAS_ELT` \*B, const `size_t` incb, `FFLAS_ELT` \*C, const `size_t` incc)
- template `INST_OR_DECL` void `fadd` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` N, const `FFLAS_ELT` \*A, const `size_t` inca, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*B, const `size_t` incb, `FFLAS_ELT` \*C, const `size_t` incc)
- template `INST_OR_DECL` void `fassign` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` m, const `size_t` n, const `FFLAS_ELT` \*B, const `size_t` ldb, `FFLAS_ELT` \*A, const `size_t` lda)  
 $fassign : A \leftarrow B.$
- template `INST_OR_DECL` void `fzero` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` m, const `size_t` n, `FFLAS_ELT` \*A, const `size_t` lda)  
 $fzero : A \leftarrow 0.$

- template `INST_OR_DECL` bool `fequal` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, const `FFLAS_ELT` \*A, const size\_t lda, const `FFLAS_ELT` \*B, const size\_t ldb)  

$$fequal : test A = B.$$
- template `INST_OR_DECL` bool `fiszero` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, const `FFLAS_ELT` \*A, const size\_t lda)  

$$fiszero : test A = 0.$$
- template `INST_OR_DECL` void `fidentity` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, `FFLAS_ELT` \*A, const size\_t lda, const `FFLAS_ELT` &d)  

$$creates a diagonal matrix$$
- template `INST_OR_DECL` void `fidentity` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, `FFLAS_ELT` \*A, const size\_t lda)  

$$creates a diagonal matrix$$
- template `INST_OR_DECL` void `freduce` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, `FFLAS_ELT` \*A, const size\_t lda)  

$$freduce A \leftarrow A mod F.$$
- template `INST_OR_DECL` void `freduce` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*A, const size\_t lda)  

$$freduce A \leftarrow B mod F.$$
- template `INST_OR_DECL` void `finit` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*A, const size\_t lda)  

$$finit A \leftarrow B mod F.$$
- template `INST_OR_DECL` void `fnegin` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, `FFLAS_ELT` \*A, const size\_t lda)  

$$fnegin A \leftarrow -A.$$
- template `INST_OR_DECL` void `fneg` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*A, const size\_t lda)  

$$fneg A \leftarrow -B.$$
- template `INST_OR_DECL` void `fscal` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, const `FFLAS_ELT` alpha, `FFLAS_ELT` \*A, const size\_t lda)  

$$fscal A \leftarrow a \cdot A.$$
- template `INST_OR_DECL` void `fscal` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*A, const size\_t lda, `FFLAS_ELT` \*B, const size\_t ldb)  

$$fscal B \leftarrow a \cdot A.$$
- template `INST_OR_DECL` void `faxpy` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*X, const size\_t idx, `FFLAS_ELT` \*Y, const size\_t ldy)  

$$faxpy : y \leftarrow \alpha \cdot x + y.$$
- template `INST_OR_DECL` void `fmove` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, `FFLAS_ELT` \*A, const size\_t lda, `FFLAS_ELT` \*B, const size\_t ldb)  

$$faxpy : y \leftarrow \alpha \cdot x + \beta \cdot y.$$
- template `INST_OR_DECL` void `fadd` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, const `FFLAS_ELT` \*A, const size\_t lda, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*C, const size\_t ldc)  

$$fadd : matrix addition.$$
- template `INST_OR_DECL` void `fsub` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, const `FFLAS_ELT` \*A, const size\_t lda, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*C, const size\_t ldc)  

$$fsub : matrix subtraction.$$
- template `INST_OR_DECL` void `fsubin` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*C, const size\_t ldc)  

$$fsubin C = C - B$$
- template `INST_OR_DECL` void `fadd` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, const `FFLAS_ELT` \*A, const size\_t lda, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*C, const size\_t ldc)

*fadd* : matrix addition with scaling.

- template `INST_OR_DECL` void `faddin` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*C, const size\_t ldc)

*faddin*

- template `INST_OR_DECL` `FFLAS_ELT` \* `fgemv` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS_TRANSPOSE` TransA, const size\_t M, const size\_t N, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*A, const size\_t lda, const `FFLAS_ELT` \*X, const size\_t incX, const `FFLAS_ELT` beta, `FFLAS_ELT` \*Y, const size\_t incY)

*finite prime FFLAS\_FIELD*<*FFLAS\_ELT*> *GEneral Matrix Vector multiplication.*

- template `INST_OR_DECL` void `fger` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*x, const size\_t incx, const `FFLAS_ELT` \*y, const size\_t incy, `FFLAS_ELT` \*A, const size\_t lda)

*fger*: rank one update of a general matrix

- template `INST_OR_DECL` void `ftsv` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS_UPLO` Uplo, const `FFLAS_TRANSPOSE` TransA, const `FFLAS_DIAG` Diag, const size\_t N, const `FFLAS_ELT` \*A, const size\_t lda, `FFLAS_ELT` \*X, int incX)

*ftsv*: *TRI*angular System solve with Vector Computes  $X \leftarrow \text{op}(A^{-1})X$

- template `INST_OR_DECL` void `ftdsm` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS_SIDE` Side, const `FFLAS_UPLO` Uplo, const `FFLAS_TRANSPOSE` TransA, const `FFLAS_DIAG` Diag, const size\_t M, const size\_t N, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*A, const size\_t lda, `FFLAS_ELT` \*B, const size\_t ldb)

*ftdsm*: *TRI*angular System solve with *Matrix*.

- template `INST_OR_DECL` void `ftmm` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS_SIDE` Side, const `FFLAS_UPLO` Uplo, const `FFLAS_TRANSPOSE` TransA, const `FFLAS_DIAG` Diag, const size\_t M, const size\_t N, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*A, const size\_t lda, `FFLAS_ELT` \*B, const size\_t ldb)

*ftmm*: *TRI*angular *Matrix Multiply*.

- template `INST_OR_DECL` `FFLAS_ELT` \* `fgemm` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS_TRANSPOSE` ta, const `FFLAS_TRANSPOSE` tb, const size\_t m, const size\_t n, const size\_t k, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*A, const size\_t lda, const `FFLAS_ELT` \*B, const size\_t ldb, const `FFLAS_ELT` beta, `FFLAS_ELT` \*C, const size\_t ldc)

*fgemm*: *Field GEneral Matrix Multiply*.

- template `INST_OR_DECL` `FFLAS_ELT` \* `fgemm` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS_TRANSPOSE` ta, const `FFLAS_TRANSPOSE` tb, const size\_t m, const size\_t n, const size\_t k, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*A, const size\_t lda, const `FFLAS_ELT` \*B, const size\_t ldb, const `FFLAS_ELT` beta, `FFLAS_ELT` \*C, const size\_t ldc, const `ParSeqHelper::Sequential` seq)
- template `INST_OR_DECL` `FFLAS_ELT` \* `fgemm` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS_TRANSPOSE` ta, const `FFLAS_TRANSPOSE` tb, const size\_t m, const size\_t n, const size\_t k, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*A, const size\_t lda, const `FFLAS_ELT` \*B, const size\_t ldb, const `FFLAS_ELT` beta, `FFLAS_ELT` \*C, const size\_t ldc, const `ParSeqHelper::Parallel`< `CuttingStrategy::Recursive`, `StrategyParameter::TwoDAdaptive` > par)
- template `INST_OR_DECL` `FFLAS_ELT` \* `fgemm` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS_TRANSPOSE` ta, const `FFLAS_TRANSPOSE` tb, const size\_t m, const size\_t n, const size\_t k, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*A, const size\_t lda, const `FFLAS_ELT` \*B, const size\_t ldb, const `FFLAS_ELT` beta, `FFLAS_ELT` \*C, const size\_t ldc, const `ParSeqHelper::Parallel`< `CuttingStrategy::Block`, `StrategyParameter::Threads` > par)
- template `INST_OR_DECL` `FFLAS_ELT` \* `fsquare` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS_TRANSPOSE` ta, const size\_t n, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*A, const size\_t lda, const `FFLAS_ELT` beta, `FFLAS_ELT` \*C, const size\_t ldc)

*fsquare*: Squares a matrix.

- template<class Cut = `CuttingStrategy::Block`, class Strat = `StrategyParameter::Threads`>  
void `BlockCuts` (size\_t &RBLOCKSIZE, size\_t &CBLOCKSIZE, const size\_t m, const size\_t n, const size\_t numthreads)
- template<> void `BlockCuts`< `CuttingStrategy::Single`, `StrategyParameter::Threads` > (size\_t &RBLOCKSIZE, size\_t &CBLOCKSIZE, const size\_t m, const size\_t n, const size\_t numthreads)

- `template<> void BlockCuts< CuttingStrategy::Row, StrategyParameter::Fixed > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t numthreads)`
- `template<> void BlockCuts< CuttingStrategy::Row, StrategyParameter::Grain > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t grainsize)`
- `template<> void BlockCuts< CuttingStrategy::Block, StrategyParameter::Grain > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t grainsize)`
- `template<> void BlockCuts< CuttingStrategy::Column, StrategyParameter::Fixed > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t numthreads)`
- `template<> void BlockCuts< CuttingStrategy::Column, StrategyParameter::Grain > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t grainsize)`
- `template<> void BlockCuts< CuttingStrategy::Block, StrategyParameter::Fixed > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t numthreads)`
- `template<> void BlockCuts< CuttingStrategy::Row, StrategyParameter::Threads > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t numthreads)`
- `template<> void BlockCuts< CuttingStrategy::Column, StrategyParameter::Threads > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t numthreads)`
- `template<> void BlockCuts< CuttingStrategy::Block, StrategyParameter::Threads > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t numthreads)`
- `template<class Cut = CuttingStrategy::Block, class Param = StrategyParameter::Threads>  
void BlockCuts (size_t &rowBlockSize, size_t &colBlockSize, size_t &lastRBS, size_t &lastCBS, size_t &changeRBS, size_t &changeCBS, size_t &numRowBlock, size_t &numColBlock, size_t m, size_t n, const size_t numthreads)`
- `template<class Field >  
void pfzero (const Field &F, size_t m, size_t n, typename Field::Element_ptr C, size_t BS=0)`
- `template<class Field, class RandIter >  
void pfrand (const Field &F, RandIter &G, size_t m, size_t n, typename Field::Element_ptr C, size_t BS=0)`
- `template<class Field, class Cut, class Param >  
Field::Element & fdot (const Field &F, const size_t N, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, typename Field::Element &d, const ParSeqHelper::Parallel< Cut, Param > par)`
- `template<class Field, class AlgoT, class FieldTrait >  
Field::Element * pfgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename Field::ConstElement_ptr A, const size_t lda, const typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element *C, const size_t ldc, MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Block, StrategyParameter::Threads > > &H)`
- `template<class Field, class AlgoT, class FieldTrait >  
Field::Element * pfgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename Field::ConstElement_ptr AA, const size_t lda, const typename Field::ConstElement_ptr BB, const size_t ldb, const typename Field::Element beta, typename Field::Element *C, const size_t ldc, MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive, StrategyParameter::ThreeDAdaptive > > &H)`
- `template<class Field, class AlgoT, class FieldTrait >  
Field::Element * pfgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename Field::ConstElement_ptr AA, const size_t lda, const typename Field::ConstElement_ptr BB, const size_t ldb, const typename Field::Element beta, typename Field::Element *C, const size_t ldc, MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive, StrategyParameter::TwoDAdaptive > > &H)`
- `template<class Field, class AlgoT, class FieldTrait >  
Field::Element * pfgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename Field::ConstElement_ptr AA, const size_t lda, const typename Field::ConstElement_ptr BB, const size_t ldb, const typename Field::Element beta, typename Field::Element *C, const size_t ldc, MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive, StrategyParameter::TwoD > > &H)`

- `template<class Field, class AlgoT, class FieldTrait >`  
`Field::Element_ptr pfgemm` (const `Field` &F, const `FFLAS_TRANSPOSE` ta, const `FFLAS_TRANSPOSE` tb, const `size_t` m, const `size_t` n, const `size_t` k, const typename `Field::Element` alpha, const typename `Field::ConstElement_ptr` A, const `size_t` lda, const typename `Field::ConstElement_ptr` B, const `size_t` ldb, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, `MMHelper`< `Field`, `AlgoT`, `FieldTrait`, `ParSeqHelper::Parallel`< `CuttingStrategy::Recursive`, `StrategyParameter::ThreeD` > > &H)
- `template<class Field, class AlgoT, class FieldTrait >`  
`Field::Element * pfgemm` (const `Field` &F, const `FFLAS_TRANSPOSE` ta, const `FFLAS_TRANSPOSE` tb, const `size_t` m, const `size_t` n, const `size_t` k, const typename `Field::Element` alpha, const typename `Field::ConstElement_ptr` A, const `size_t` lda, const typename `Field::ConstElement_ptr` B, const `size_t` ldb, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, `MMHelper`< `Field`, `AlgoT`, `FieldTrait`, `ParSeqHelper::Parallel`< `CuttingStrategy::Recursive`, `StrategyParameter::ThreeDInPlace` > > &H)
- `template<class Field, class AlgoT, class FieldTrait >`  
`Field::Element_ptr fgemv` (const `Field` &F, const `FFLAS_TRANSPOSE` ta, const `size_t` m, const `size_t` n, const typename `Field::Element` alpha, const typename `Field::ConstElement_ptr` A, const `size_t` lda, const typename `Field::ConstElement_ptr` X, const `size_t` incX, const typename `Field::Element` beta, typename `Field::Element_ptr` Y, const `size_t` incY, `MMHelper`< `Field`, `AlgoT`, `FieldTrait`, `ParSeqHelper::Parallel`< `CuttingStrategy::Recursive`, `StrategyParameter::Threads` > > &H)
- `template<class Field, class AlgoT, class FieldTrait, class Cut >`  
`Field::Element_ptr fgemv` (const `Field` &F, const `FFLAS_TRANSPOSE` ta, const `size_t` m, const `size_t` n, const typename `Field::Element` alpha, const typename `Field::ConstElement_ptr` A, const `size_t` lda, const typename `Field::ConstElement_ptr` X, const `size_t` incX, const typename `Field::Element` beta, typename `Field::Element_ptr` Y, const `size_t` incY, `MMHelper`< `Field`, `AlgoT`, `FieldTrait`, `ParSeqHelper::Parallel`< `CuttingStrategy::Row`, `Cut` > > &H)
- `void parseArguments` (int argc, char \*\*argv, `Argument` \*args, bool printDefaults=true)
- `char * getArgumentsValue` (int argc, char \*\*argv, int i)  
*Get the value of an argument and avoid core dump when no value was given after an argument.*
- `std::ostream & writeCommandString` (std::ostream &os, `Argument` \*args, const char \*programName=nullptr)  
*writes the values of all arguments, preceded by the programName*
- `template<class Field >`  
`std::ostream & WriteMatrix` (std::ostream &c, const `Field` &F, `size_t` m, `size_t` n, typename `Field::ConstElement_ptr` A, `size_t` lda, `FFLAS_FORMAT` format, bool column\_major)  
*WriteMatrix: write a matrix to an output stream.*
- `void preamble` (std::ifstream &ifs, `FFLAS_FORMAT` &format)
- `template<class Field >`  
`Field::Element_ptr ReadMatrix` (std::ifstream &ifs, `Field` &F, `size_t` &m, `size_t` &n, typename `Field::Element_ptr` &A, `FFLAS_FORMAT` format=`FflasAuto`)  
*ReadMatrix: read a matrix from an input stream.*
- `template<class Field >`  
`Field::Element_ptr ReadMatrix` (const std::string &matrix\_file, `Field` &F, `size_t` &m, `size_t` &n, typename `Field::Element_ptr` &A, `FFLAS_FORMAT` format=`FflasAuto`)  
*ReadMatrix: read a matrix from a file.*
- `template<class Field >`  
`void WriteMatrix` (std::string &matrix\_file, const `Field` &F, int m, int n, typename `Field::ConstElement_ptr` A, `size_t` lda, `FFLAS_FORMAT` format=`FflasDense`, bool column\_major=false)  
*WriteMatrix: write a matrix to a file.*
- `std::ostream & WritePermutation` (std::ostream &c, const `size_t` \*P, `size_t` N)  
*WritePermutation: write a permutation matrix to an output stream.*
- `template<class Element >`  
`bool alignable` ()
- `template<> bool alignable`< `Givaro::Integer` \* > ()
- `template<class Field >`  
`Field::Element_ptr fflas_new` (const `Field` &F, const `size_t` m, const `Alignment` align=`Alignment::DEFAULT`)

- `template<class Field >`  
`Field::Element_ptr fflas_new` (const Field &F, const size\_t m, const size\_t n, const Alignment align=Alignment::DEFAULT)
- `template<class Element >`  
`Element * fflas_new` (const size\_t m, const Alignment align=Alignment::DEFAULT)
- `template<class Element_ptr >`  
`void fflas_delete` (Element\_ptr A)
- `template<class Ptr, class ... Args>`  
`void fflas_delete` (Ptr p, Args ... args)
- `void prefetch` (const int64\_t \*)
- `void getTLBSize` (int &tlb)
- `void queryCacheSizes` (int &l1, int &l2, int &l3)
- `int queryL1CacheSize` ()
- `int queryTopLevelCacheSize` ()
- `uint64_t getSeed` ()

### 11.1.1 Typedef Documentation

#### 11.1.1.1 Checker\_fgemm

```
template<class Field >
using Checker_fgemm = FFLAS::Checker_Empty<Field>
```

#### 11.1.1.2 Checker\_ftrsm

```
template<class Field >
using Checker_ftrsm = FFLAS::Checker_Empty<Field>
```

#### 11.1.1.3 ForceCheck\_fgemm

```
template<class Field >
using ForceCheck_fgemm = CheckerImplem_fgemm<Field>
```

#### 11.1.1.4 ForceCheck\_ftrsm

```
template<class Field >
using ForceCheck_ftrsm = CheckerImplem_ftrsm<Field>
```

#### 11.1.1.5 ZOSparseMatrix

```
using ZOSparseMatrix = std::true_type
```

#### 11.1.1.6 NotZOSparseMatrix

```
using NotZOSparseMatrix = std::false_type
```

#### 11.1.1.7 SimdSparseMatrix

```
using SimdSparseMatrix = std::true_type
```

#### 11.1.1.8 NoSimdSparseMatrix

```
using NoSimdSparseMatrix = std::false_type
```

#### 11.1.1.9 MKLSparseMatrixFormat

```
using MKLSparseMatrixFormat = std::true_type
```

#### 11.1.1.10 NotMKLSparseMatrixFormat

```
using NotMKLSparseMatrixFormat = std::false_type
```

#### 11.1.1.11 has\_plus

```
template<class T >
using has_plus = typename std::conditional<std::is_arithmetic<T>::value, std::true_type,
has_plus_impl<T>>::type
```

#### 11.1.1.12 has\_minus

```
template<class T >
using has_minus = typename std::conditional<std::is_arithmetic<T>::value, std::true_type,
has_minus_impl<T>>::type
```

#### 11.1.1.13 has\_equal

```
template<class T >
using has_equal = typename std::conditional<std::is_arithmetic<T>::value, std::true_type,
std::is_copy_assignable<T>>::type
```

#### 11.1.1.14 has\_plus\_eq

```
template<class T >
using has_plus_eq = typename std::conditional<std::is_arithmetic<T>::value, std::true_type,
has_plus_eq_impl<T>>::type
```

#### 11.1.1.15 has\_minus\_eq

```
template<class T >
using has_minus_eq = typename std::conditional<std::is_arithmetic<T>::value, std::true_type,
has_minus_eq_impl<T>>::type
```

**11.1.1.16 has\_mul**

```
template<class T >
using has_mul = typename std::conditional<std::is_arithmetic<T>::value, std::true_type, has_mul_impl<T>>::type
```

**11.1.1.17 has\_mul\_eq**

```
template<class T >
using has_mul_eq = typename std::conditional<std::is_arithmetic<T>::value, std::true_type, has_mul_eq_impl<T>>::type
```

**11.1.1.18 Timer**

```
typedef Givaro::Timer Timer
```

**11.1.1.19 BaseTimer**

```
typedef Givaro::BaseTimer BaseTimer
```

**11.1.1.20 UserTimer**

```
typedef Givaro::UserTimer UserTimer
```

**11.1.1.21 SysTimer**

```
typedef Givaro::SysTimer SysTimer
```

**11.1.2 Enumeration Type Documentation****11.1.2.1 FFLAS\_ORDER**

```
enum FFLAS_ORDER
```

Storage by row or col ?

Enumerator

FflasRowMajor	row major
FflasColMajor	col major

**11.1.2.2 FFLAS\_TRANSPOSE**

```
enum FFLAS_TRANSPOSE
```

Is matrix transposed ?



## Enumerator

FflasNoTrans	Matrix is not transposed.
FflasTrans	Matrix is transposed.

## 11.1.2.3 FFLAS\_UPLO

enum FFLAS\_UPLO

Is triangular matrix's shape upper ?

## Enumerator

FflasUpper	Triangular matrix is Upper triangular (if $i > j$ then $T_{i,j} = 0$ )
FflasLower	Triangular matrix is Lower triangular (if $i < j$ then $T_{i,j} = 0$ )
FflasLeftTri	Triangular matrix is Left triangular (if $j > n - i - 1$ then $T_{i,j} = 0$ )
FflasRightTri	Triangular matrix is Right triangular (if $j < n - i - 1$ then $T_{i,j} = 0$ )

## 11.1.2.4 FFLAS\_DIAG

enum FFLAS\_DIAG

Is the triangular matrix implicitly unit diagonal ?

## Enumerator

FflasNonUnit	Triangular matrix has an explicit arbitrary diagonal.
FflasUnit	Triangular matrix has an implicit unit diagonal ( $T_{i,i} = 1$ )

## 11.1.2.5 FFLAS\_SIDE

enum FFLAS\_SIDE

On what side ?

## Enumerator

FflasLeft	Operator applied on the left.
FflasRight	Operator applied on the right.

## 11.1.2.6 FFLAS\_BASE

enum FFLAS\_BASE

FFLAS\_BASE determines the type of the element representation for Matrix Mult kernel.

(deprecated, should not be used)

## Enumerator

FflasDouble	to use the double precision BLAS
FflasFloat	to use the single precision BLAS
FflasGeneric	for any other domain, that can not be converted to floating point integers

## 11.1.2.7 number\_kind

```
enum number_kind
```

## Enumerator

zero	
one	
mone	
other	

## 11.1.2.8 SparseMatrix\_t

```
enum class SparseMatrix_t [strong]
```

## Enumerator

CSR	
CSR_ZO	
CSC	
CSC_ZO	
COO	
COO_ZO	
ELL	
ELL_ZO	
SELL	
SELL_ZO	
ELL_simd	
ELL_simd_ZO	
CSR_HYB	
HYB_ZO	

## 11.1.2.9 FFLAS\_FORMAT

```
enum FFLAS_FORMAT
```

## Enumerator

FflasAuto	
FflasDense	
FflasSMS	
FflasBinary	
FflasMath	
FflasMaple	
FflasSageMath	

### 11.1.3 Function Documentation

#### 11.1.3.1 InfNorm()

```
Givaro::Integer InfNorm (  
    const size_t M,  
    const size_t N,  
    const Givaro::Integer * A,  
    const size_t lda) [inline]
```

#### 11.1.3.2 min3()

```
template<class T >  
const T & min3 (  
    const T & m,  
    const T & n,  
    const T & k)
```

#### 11.1.3.3 max3()

```
template<class T >  
const T & max3 (  
    const T & m,  
    const T & n,  
    const T & k)
```

#### 11.1.3.4 min4()

```
template<class T >  
const T & min4 (  
    const T & m,  
    const T & n,  
    const T & k,  
    const T & l)
```

#### 11.1.3.5 max4()

```
template<class T >  
const T & max4 (  
    const T & m,  
    const T & n,  
    const T & k,  
    const T & l)
```

#### 11.1.3.6 fadd() [1/8]

```
template<class Field >
void fadd (
    const Field & F,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t inca,
    typename Field::ConstElement_ptr B,
    const size_t incb,
    typename Field::Element_ptr C,
    const size_t incc)
```

#### 11.1.3.7 faddin() [1/5]

```
template<class Field >
void faddin (
    const Field & F,
    const size_t N,
    typename Field::ConstElement_ptr B,
    const size_t incb,
    typename Field::Element_ptr C,
    const size_t incc)
```

#### 11.1.3.8 fsub() [1/4]

```
template<class Field >
void fsub (
    const Field & F,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t inca,
    typename Field::ConstElement_ptr B,
    const size_t incb,
    typename Field::Element_ptr C,
    const size_t incc)
```

#### 11.1.3.9 fsubin() [1/3]

```
template<class Field >
void fsubin (
    const Field & F,
    const size_t N,
    typename Field::ConstElement_ptr B,
    const size_t incb,
    typename Field::Element_ptr C,
    const size_t incc)
```

### 11.1.3.10 fadd() [2/8]

```
template<class Field >
void fadd (
    const Field & F,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t inca,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr B,
    const size_t incb,
    typename Field::Element_ptr C,
    const size_t incc)
```

**Todo** optimise here

### 11.1.3.11 pfadd()

```
template<class Field >
void pfadd (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    typename Field::Element_ptr C,
    const size_t ldc,
    const size_t numths)
```

### 11.1.3.12 pfsub()

```
template<class Field >
void pfsub (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    typename Field::Element_ptr C,
    const size_t ldc,
    const size_t numths)
```

### 11.1.3.13 pfaddin()

```
template<class Field >
void pfaddin (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    typename Field::Element_ptr C,
    const size_t ldc,
    size_t numths)
```

### 11.1.3.14 pfsubin()

```
template<class Field >
void pfsubin (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    typename Field::Element_ptr C,
    const size_t ldc,
    size_t numths)
```

### 11.1.3.15 fadd() [3/8]

```
template<class Field >
void fadd (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    typename Field::Element_ptr C,
    const size_t ldc)
```

fadd : matrix addition.

Computes  $C = A + B$ .

#### Parameters

<i>F</i>	field
<i>M</i>	rows
<i>N</i>	cols
<i>A</i>	dense matrix of size MxN
<i>lda</i>	leading dimension of A
<i>B</i>	dense matrix of size MxN
<i>ldb</i>	leading dimension of B
<i>C</i>	dense matrix of size MxN
<i>ldc</i>	leading dimension of C

**11.1.3.16 fsub()** [2/4]

```
template<class Field >
void fsub (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    typename Field::Element_ptr C,
    const size_t ldc)
```

fsub : matrix subtraction.

Computes  $C = A - B$ .

**Parameters**

<i>F</i>	field
<i>M</i>	rows
<i>N</i>	cols
<i>A</i>	dense matrix of size MxN
<i>lda</i>	leading dimension of A
<i>B</i>	dense matrix of size MxN
<i>ldb</i>	leading dimension of B
<i>C</i>	dense matrix of size MxN
<i>ldc</i>	leading dimension of C

**11.1.3.17 faddin()** [2/5]

```
template<class Field >
void faddin (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    typename Field::Element_ptr C,
    const size_t ldc)
```

faddin

**11.1.3.18 faddin()** [3/5]

```
template<class Field >
void faddin (
    const Field & F,
    const FFLAS_UPLO uplo,
    const size_t N,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    typename Field::Element_ptr C,
    const size_t ldc)
```

fadding for symmetric matrices

**11.1.3.19 fsubin()** [2/3]

```
template<class Field >
void fsubin (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    typename Field::Element_ptr C,
    const size_t ldc)
```

fsubin C = C - B

**11.1.3.20 fadd()** [4/8]

```
template<class Field >
void fadd (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    typename Field::Element_ptr C,
    const size_t ldc)
```

fadd : matrix addition with scaling.

Computes  $C = A + \text{alpha } B$ .

**Parameters**

<i>F</i>	field
<i>M</i>	rows
<i>N</i>	cols
<i>A</i>	dense matrix of size MxN
<i>lda</i>	leading dimension of A
<i>alpha</i>	some scalar
<i>B</i>	dense matrix of size MxN
<i>ldb</i>	leading dimension of B
<i>C</i>	dense matrix of size MxN
<i>ldc</i>	leading dimension of C

**11.1.3.21 fassign()** [1/10]

```
template<class Field >
void fassign (
    const Field & F,
    const size_t N,
```



```

typename Field::ConstElement_ptr Y,
const size_t incY,
typename Field::Element_ptr X,
const size_t incX) [inline]

```

fassign :  $x \leftarrow y$ .

X is preallocated

**Todo** variant for triangular matrix

#### Parameters

	$F$	field
	$N$	size of the vectors
out	$X$	vector in $F$
	$incX$	stride of X
in	$Y$	vector in $F$
	$incY$	stride of Y

#### 11.1.3.22 fassign() [2/10]

```

template<>
void fassign (
    const Givaro::Modular< float > & F,
    const size_t N,
    const float * Y,
    const size_t incY,
    float * X,
    const size_t incX) [inline]

```

#### 11.1.3.23 fassign() [3/10]

```

template<>
void fassign (
    const Givaro::ModularBalanced< float > & F,
    const size_t N,
    const float * Y,
    const size_t incY,
    float * X,
    const size_t incX) [inline]

```

#### 11.1.3.24 fassign() [4/10]

```

template<>
void fassign (
    const Givaro::ZRing< float > & F,
    const size_t N,
    const float * Y,
    const size_t incY,
    float * X,
    const size_t incX) [inline]

```

**11.1.3.25 fassign()** [5/10]

```
template<>
void fassign (
    const Givaro::Modular< double > & F,
    const size_t N,
    const double * Y,
    const size_t incY,
    double * X,
    const size_t incX) [inline]
```

**11.1.3.26 fassign()** [6/10]

```
template<>
void fassign (
    const Givaro::ModularBalanced< double > & F,
    const size_t N,
    const double * Y,
    const size_t incY,
    double * X,
    const size_t incX) [inline]
```

**11.1.3.27 fassign()** [7/10]

```
template<>
void fassign (
    const Givaro::ZRing< double > & F,
    const size_t N,
    const double * Y,
    const size_t incY,
    double * X,
    const size_t incX) [inline]
```

**11.1.3.28 fassign()** [8/10]

```
template<class Field >
void fassign (
    const Field & F,
    const size_t m,
    const size_t n,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    typename Field::Element_ptr A,
    const size_t lda)
```

fassign :  $A \leftarrow B$ .

**Parameters**

$F$	field
$m$	number of rows to copy
$n$	number of cols to copy
$A$	matrix in F
$lda$	stride of A
$B$	vector in F
$ldb$	stride of B

**11.1.3.29 faxpy()** [1/6]

```
template<class Field >
void faxpy (
    const Field & F,
    const size_t N,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr X,
    const size_t incX,
    typename Field::Element_ptr Y,
    const size_t incY) [inline]
```

$\text{faxpy} : y \leftarrow \alpha \cdot x + y.$

**Parameters**

	$F$	field
	$N$	size of the vectors
	$\alpha$	scalar
in	$X$	vector in F
	$\text{incX}$	stride of X
in, out	$Y$	vector in F
	$\text{incY}$	stride of Y

**11.1.3.30 faxpy()** [2/6]

```
template<>
void faxpy (
    const Givaro::DoubleDomain & ,
    const size_t N,
    const Givaro::DoubleDomain::Element a,
    Givaro::DoubleDomain::ConstElement_ptr x,
    const size_t incx,
    Givaro::DoubleDomain::Element_ptr y,
    const size_t incy) [inline]
```

**11.1.3.31 faxpy()** [3/6]

```
template<>
void faxpy (
    const Givaro::FloatDomain & ,
    const size_t N,
    const Givaro::FloatDomain::Element a,
    Givaro::FloatDomain::ConstElement_ptr x,
    const size_t incx,
    Givaro::FloatDomain::Element_ptr y,
    const size_t incy) [inline]
```

### 11.1.3.32 faxpy() [4/6]

```
template<class Field >
void faxpy (
    const Field & F,
    const size_t m,
    const size_t n,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr X,
    const size_t ldx,
    typename Field::Element_ptr Y,
    const size_t ldy) [inline]
```

$\text{faxpy} : y \leftarrow \alpha \cdot x + y.$

#### Parameters

	$F$	field
	$m$	row dimension
	$n$	column dimension
	$\alpha$	scalar
in	$X$	vector in $F$
	$ldx$	leading dimension of $X$
in, out	$Y$	vector in $F$
	$ldy$	leading dimension of $Y$

### 11.1.3.33 fdot() [1/11]

```
template<class Field >
Field::Element fdot (
    const Field & F,
    const size_t N,
    typename Field::ConstElement_ptr x,
    const size_t incx,
    typename Field::ConstElement_ptr y,
    const size_t incy,
    ModeCategories::DefaultTag & MT) [inline]
```

### 11.1.3.34 fdot() [2/11]

```
template<class Field >
Field::Element fdot (
    const Field & F,
    const size_t N,
    typename Field::ConstElement_ptr x,
    const size_t incx,
    typename Field::ConstElement_ptr y,
    const size_t incy,
    ModeCategories::DelayedTag & MT) [inline]
```

**11.1.3.35 fdot()** [3/11]

```
template<>
Givaro::DoubleDomain::Element fdot (
    const Givaro::DoubleDomain & ,
    const size_t N,
    Givaro::DoubleDomain::ConstElement_ptr x,
    const size_t incx,
    Givaro::DoubleDomain::ConstElement_ptr y,
    const size_t incy,
    ModeCategories::DefaultTag & MT) [inline]
```

**11.1.3.36 fdot()** [4/11]

```
template<>
Givaro::FloatDomain::Element fdot (
    const Givaro::FloatDomain & ,
    const size_t N,
    Givaro::FloatDomain::ConstElement_ptr x,
    const size_t incx,
    Givaro::FloatDomain::ConstElement_ptr y,
    const size_t incy,
    ModeCategories::DefaultTag & MT) [inline]
```

**11.1.3.37 fdot()** [5/11]

```
template<class Field , class T >
Field::Element fdot (
    const Field & F,
    const size_t N,
    typename Field::ConstElement_ptr x,
    const size_t incx,
    typename Field::ConstElement_ptr y,
    const size_t incy,
    ModeCategories::ConvertTo< T > & MT) [inline]
```

**11.1.3.38 fdot()** [6/11]

```
template<class Field >
Field::Element fdot (
    const Field & F,
    const size_t N,
    typename Field::ConstElement_ptr x,
    const size_t incx,
    typename Field::ConstElement_ptr y,
    const size_t incy,
    ModeCategories::DefaultBoundedTag & dbt) [inline]
```

**11.1.3.39 fdot()** [7/11]

```
template<class Field >
Field::Element fdot (
    const Field & F,
    const size_t N,
    typename Field::ConstElement_ptr x,
    const size_t incx,
    typename Field::ConstElement_ptr y,
    const size_t incy,
    const ParSeqHelper::Sequential seq) [inline]
```

**11.1.3.40 fdot()** [8/11]

```
template<class Field >
Field::Element fdot (
    const Field & F,
    const size_t N,
    typename Field::ConstElement_ptr X,
    const size_t incX,
    typename Field::ConstElement_ptr Y,
    const size_t incY) [inline]
```

fdot: dot product  $x^T y$ .

**Parameters**

$F$	field
$N$	size of the vectors
$X$	vector in F
$incX$	stride of X
$Y$	vector in F
$incY$	stride of Y

**11.1.3.41 fgemm()** [1/23]

```
template<class Field >
Field::Element_ptr fgemm (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::Winograd, ModeCategories::ConvertTo< ElementCategories::MachineFl
>, ParSeqHelper::Sequential > & H) [inline]
```

**11.1.3.42 fgemm()** [2/23]

```

template<typename Field >
Field::Element_ptr fgemm (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    const ParSeqHelper::Sequential seq) [inline]

```

**11.1.3.43 fgemm()** [3/23]

```

template<typename Field , class Cut , class Param >
Field::Element_ptr fgemm (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    const ParSeqHelper::Parallel< Cut, Param > par) [inline]

```

**11.1.3.44 fgemm()** [4/23]

```

template<typename Field >
Field::Element_ptr fgemm (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,

```

```

const size_t ldb,
const typename Field::Element beta,
typename Field::Element_ptr C,
const size_t ldc) [inline]

```

**fgemm**: Field **GE**neral **M**atrix **M**ultiply.

Computes  $C = \alpha \text{op}(A) \times \text{op}(B) + \beta C$  Automatically set Winograd recursion level

#### Parameters

<i>F</i>	field.
<i>ta</i>	if $ta == \text{FflaSTrans}$ then $\text{op}(A) = A^t$ , else $\text{op}(A) = A$ ,
<i>tb</i>	same for matrix B
<i>m</i>	see A
<i>n</i>	see B
<i>k</i>	see A
<i>alpha</i>	scalar
<i>beta</i>	scalar
<i>A</i>	$\text{op}(A)$ is $m \times k$
<i>B</i>	$\text{op}(B)$ is $k \times n$
<i>C</i>	$C$ is $m \times n$
<i>lda</i>	leading dimension of A
<i>ldb</i>	leading dimension of B
<i>ldc</i>	leading dimension of C
<i>w</i>	recursive levels of Winograd's algorithm are used. No argument (or -1) does auto computation of $w$ .

#### Warning

$\alpha$  must be invertible

#### 11.1.3.45 fgemm() [5/23]

```

template<typename Field , class ModeT , class ParSeq >
Field::Element_ptr fgemm (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::Auto, ModeT, ParSeq > & H) [inline]

```



**11.1.3.46 fgemm()** [6/23]

```

template<class Field >
Field::Element_ptr fgemm (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::Winograd, ModeCategories::DelayedTag, ParSeqHelper::Sequential
> & H) [inline]

```

**11.1.3.47 fsquare()** [1/6]

```

template<class Field >
Field::Element_ptr fsquare (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const size_t n,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc) [inline]

```

fsquare: Squares a matrix.

compute  $C \leftarrow \alpha \text{op}(A) \text{op}(A) + \beta C$  over a Field F Avoid the conversion of B

**Parameters**

<i>ta</i>	if $ta == \text{FflasTrans}$ , $\text{op}(A) = A^T$ .
<i>F</i>	field
<i>n</i>	size of A
<i>alpha</i>	scalar
<i>beta</i>	scalar
<i>A</i>	dense matrix of size $n \times n$
<i>lda</i>	leading dimension of A
<i>C</i>	dense matrix of size $n \times n$
<i>ldc</i>	leading dimension of C

**Bug** why double ?

**11.1.3.48 fsquare()** [2/6]

```
template<>
double * fsquare (
    const Givaro::ModularBalanced< double > & F,
    const FFLAS_TRANSPOSE ta,
    const size_t n,
    const double alpha,
    const double * A,
    const size_t lda,
    const double beta,
    double * C,
    const size_t ldc) [inline]
```

**11.1.3.49 fsquare()** [3/6]

```
template<>
float * fsquare (
    const Givaro::ModularBalanced< float > & F,
    const FFLAS_TRANSPOSE ta,
    const size_t n,
    const float alpha,
    const float * A,
    const size_t lda,
    const float beta,
    float * C,
    const size_t ldc) [inline]
```

**11.1.3.50 fsquare()** [4/6]

```
template<>
double * fsquare (
    const Givaro::Modular< double > & F,
    const FFLAS_TRANSPOSE ta,
    const size_t n,
    const double alpha,
    const double * A,
    const size_t lda,
    const double beta,
    double * C,
    const size_t ldc) [inline]
```

**11.1.3.51 fsquare()** [5/6]

```
template<>
float * fsquare (
    const Givaro::Modular< float > & F,
    const FFLAS_TRANSPOSE ta,
    const size_t n,
    const float alpha,
    const float * A,
    const size_t lda,
    const float beta,
    float * C,
    const size_t ldc) [inline]
```

**11.1.3.52 fgemm()** [7/23]

```

template<typename RNS , typename ParSeqTrait >
FFPACK::RNSInteger< RNS >::Element_ptr fgemm (
    const FFPACK::RNSInteger< RNS > & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename FFPACK::RNSInteger< RNS >::Element alpha,
    typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Ad,
    const size_t lda,
    typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Bd,
    const size_t ldb,
    const typename FFPACK::RNSInteger< RNS >::Element beta,
    typename FFPACK::RNSInteger< RNS >::Element_ptr Cd,
    const size_t ldc,
    MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag,
    ParSeqHelper::Compose< ParSeqHelper::Sequential, ParSeqTrait > > & H) [inline]

```

**11.1.3.53 fgemm()** [8/23]

```

template<typename RNS >
FFPACK::RNSInteger< RNS >::Element_ptr fgemm (
    const FFPACK::RNSInteger< RNS > & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename FFPACK::RNSInteger< RNS >::Element alpha,
    typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Ad,
    const size_t lda,
    typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Bd,
    const size_t ldb,
    const typename FFPACK::RNSInteger< RNS >::Element beta,
    typename FFPACK::RNSInteger< RNS >::Element_ptr Cd,
    const size_t ldc,
    MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag,
    ParSeqHelper::Sequential > & H) [inline]

```

**11.1.3.54 fgemm()** [9/23]

```

template<typename RNS , typename ParSeqTrait >
FFPACK::RNSInteger< RNS >::Element_ptr fgemm (
    const FFPACK::RNSInteger< RNS > & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename FFPACK::RNSInteger< RNS >::Element alpha,
    typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Ad,

```

```

    const size_t lda,
    typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Bd,
    const size_t ldb,
    const typename FFPACK::RNSInteger< RNS >::Element beta,
    typename FFPACK::RNSInteger< RNS >::Element_ptr Cd,
    const size_t ldc,
    MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag,
    ParSeqHelper::Compose< ParSeqHelper::Parallel< CuttingStrategy::RNSModulus, StrategyParameter::Threads
    >, ParSeqTrait > > & H) [inline]

```

### 11.1.3.55 fgemm() [10/23]

```

template<typename RNS , typename Cut , typename Param >
FFPACK::RNSInteger< RNS >::Element_ptr fgemm (
    const FFPACK::RNSInteger< RNS > & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename FFPACK::RNSInteger< RNS >::Element alpha,
    typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Ad,
    const size_t lda,
    typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Bd,
    const size_t ldb,
    const typename FFPACK::RNSInteger< RNS >::Element beta,
    typename FFPACK::RNSInteger< RNS >::Element_ptr Cd,
    const size_t ldc,
    MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag,
    ParSeqHelper::Parallel< Cut, Param > > & H) [inline]

```

### 11.1.3.56 fgemm() [11/23]

```

template<class ParSeq >
Givaro::Integer * fgemm (
    const Givaro::ZRing< Givaro::Integer > & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const Givaro::Integer alpha,
    const Givaro::Integer * A,
    const size_t lda,
    const Givaro::Integer * B,
    const size_t ldb,
    Givaro::Integer beta,
    Givaro::Integer * C,
    const size_t ldc,
    MMHelper< Givaro::ZRing< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo<
    ElementCategories::RNSElementTag >, ParSeq > & H) [inline]

```

**11.1.3.57 fgemm()** [12/23]

```

template<typename RNS , class ModeT >
RNS::Element_ptr fgemm (
    const FFPACK::RNSInteger< RNS > & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename RNS::Element alpha,
    typename RNS::ConstElement_ptr Ad,
    const size_t lda,
    typename RNS::ConstElement_ptr Bd,
    const size_t ldb,
    const typename RNS::Element beta,
    typename RNS::Element_ptr Cd,
    const size_t ldc,
    MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Winograd, ModeT, ParSeqHelper::Sequential
> & H) [inline]

```

**11.1.3.58 fgemm()** [13/23]

```

template<typename RNS >
RNS::Element_ptr fgemm (
    const FFPACK::RNSIntegerMod< RNS > & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename RNS::Element alpha,
    typename RNS::ConstElement_ptr Ad,
    const size_t lda,
    typename RNS::ConstElement_ptr Bd,
    const size_t ldb,
    const typename RNS::Element beta,
    typename RNS::Element_ptr Cd,
    const size_t ldc,
    MMHelper< FFPACK::RNSIntegerMod< RNS >, MMHelperAlgo::Winograd > & H) [inline]

```

**11.1.3.59 fgemm()** [14/23]

```

Givaro::Integer * fgemm (
    const Givaro::Modular< Givaro::Integer > & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const Givaro::Integer alpha,
    const Givaro::Integer * A,
    const size_t lda,
    const Givaro::Integer * B,

```

```

    const size_t ldb,
    const Givaro::Integer beta,
    Givaro::Integer * C,
    const size_t ldc,
    MMHelper< Givaro::Modular< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo<
ElementCategories::RNSElementTag > > & H) [inline]

```

#### 11.1.3.60 fgemm() [15/23]

```

template<class ParSeq >
Givaro::Integer * fgemm (
    const Givaro::Modular< Givaro::Integer > & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const Givaro::Integer alpha,
    const Givaro::Integer * A,
    const size_t lda,
    const Givaro::Integer * B,
    const size_t ldb,
    const Givaro::Integer beta,
    Givaro::Integer * C,
    const size_t ldc,
    MMHelper< Givaro::Modular< Givaro::Integer >, MMHelperAlgo::Auto, ModeCategories::ConvertTo<
ElementCategories::RNSElementTag >, ParSeq > & H) [inline]

```

#### 11.1.3.61 fgemm() [16/23]

```

template<size_t K1, size_t K2, class ParSeq >
RecInt::ruint< K1 > * fgemm (
    const Givaro::Modular< RecInt::ruint< K1 >, RecInt::ruint< K2 > > & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const RecInt::ruint< K1 > alpha,
    const RecInt::ruint< K1 > * A,
    const size_t lda,
    const RecInt::ruint< K1 > * B,
    const size_t ldb,
    RecInt::ruint< K1 > beta,
    RecInt::ruint< K1 > * C,
    const size_t ldc,
    MMHelper< Givaro::Modular< RecInt::ruint< K1 >, RecInt::ruint< K2 > >, MMHelperAlgo::Classic,
ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq > & H) [inline]

```

#### 11.1.3.62 fgemm() [17/23]

```

template<class Field , class ModeT >
Field::Element_ptr fgemm (

```

```

    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::Winograd, ModeT > & H) [inline]

```

### 11.1.3.63 fgemm() [18/23]

```

template<class Field , class ModeT , class Cut , class Param >
Field::Element_ptr fgemm (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::WinogradPar, ModeT, ParSeqHelper::Parallel< Cut,
Param > > & H) [inline]

```

### 11.1.3.64 fgemv() [1/19]

```

template<class Field >
Field::Element_ptr fgemv (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const size_t M,
    const size_t N,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr X,
    const size_t incX,
    const typename Field::Element beta,
    typename Field::Element_ptr Y,
    const size_t incY,
    MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::MachineFlo
> > & H) [inline]

```

**11.1.3.65 fgemv()** [2/19]

```

template<class Field >
Field::Element_ptr fgemv (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const size_t M,
    const size_t N,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr X,
    const size_t incX,
    const typename Field::Element beta,
    typename Field::Element_ptr Y,
    const size_t incY,
    MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DelayedTag > & H) [inline]

```

**11.1.3.66 fgemv()** [3/19]

```

template<class Field >
Field::Element_ptr fgemv (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const size_t M,
    const size_t N,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr X,
    const size_t incX,
    const typename Field::Element beta,
    typename Field::Element_ptr Y,
    const size_t incY,
    MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DefaultTag > & H) [inline]

```

**11.1.3.67 fgemv()** [4/19]

```

template<class Field >
Field::Element_ptr fgemv (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const size_t M,
    const size_t N,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr X,
    const size_t incX,
    const typename Field::Element beta,
    typename Field::Element_ptr Y,
    const size_t incY,
    MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::LazyTag > & H) [inline]

```



## 11.1.3.68 fgemv() [5/19]

```
template<class Field >
Field::Element_ptr fgemv (
    const Field & F,
    const FFLAS_TRANSPOSE TransA,
    const size_t M,
    const size_t N,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr X,
    const size_t incX,
    const typename Field::Element beta,
    typename Field::Element_ptr Y,
    const size_t incY) [inline]
```

finite prime Field GEneral Matrix Vector multiplication.

Computes  $Y \leftarrow \alpha \text{op}(A)X + \beta Y$ .

## Parameters

	$F$	field
	$TransA$	if $TransA == FflasTrans$ then $\text{op}(A) = A^t$ .
	$M$	rows
	$N$	cols
	$alpha$	scalar
	$A$	dense matrix of size $M \times N$
	$lda$	leading dimension of $A$
	$X$	dense vector of size $N$
	$incX$	stride of $X$
	$beta$	scalar
out	$Y$	dense vector of size $M$
	$incY$	stride of $Y$

## 11.1.3.69 fgemv() [6/19]

```
Givaro::ZRing< int64_t >::Element_ptr fgemv (
    const Givaro::ZRing< int64_t > & F,
    const FFLAS_TRANSPOSE ta,
    const size_t M,
    const size_t N,
    const int64_t alpha,
    const int64_t * A,
    const size_t lda,
    const int64_t * X,
    const size_t incX,
    const int64_t beta,
    int64_t * Y,
    const size_t incY,
    MMHelper< Givaro::ZRing< int64_t >, MMHelperAlgo::Classic, ModeCategories::DefaultTag
> & H) [inline]
```

**11.1.3.70 fgemv()** [7/19]

```
Givaro::DoubleDomain::Element_ptr fgemv (
    const Givaro::DoubleDomain & F,
    const FFLAS_TRANSPOSE ta,
    const size_t M,
    const size_t N,
    const Givaro::DoubleDomain::Element alpha,
    const Givaro::DoubleDomain::ConstElement_ptr A,
    const size_t lda,
    const Givaro::DoubleDomain::ConstElement_ptr X,
    const size_t incX,
    const Givaro::DoubleDomain::Element beta,
    Givaro::DoubleDomain::Element_ptr Y,
    const size_t incY,
    MMHelper< Givaro::DoubleDomain, MMHelperAlgo::Classic, ModeCategories::DefaultTag
> & H) [inline]
```

**11.1.3.71 fgemv()** [8/19]

```
template<class Field >
Field::Element_ptr fgemv (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const size_t M,
    const size_t N,
    const typename Field::Element alpha,
    const typename Field::ConstElement_ptr A,
    const size_t lda,
    const typename Field::ConstElement_ptr X,
    const size_t incX,
    const typename Field::Element beta,
    typename Field::Element_ptr Y,
    const size_t incY,
    MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DefaultBoundedTag > & H)
[inline]
```

**11.1.3.72 fgemv()** [9/19]

```
Givaro::FloatDomain::Element_ptr fgemv (
    const Givaro::FloatDomain & F,
    const FFLAS_TRANSPOSE ta,
    const size_t M,
    const size_t N,
    const Givaro::FloatDomain::Element alpha,
    const Givaro::FloatDomain::ConstElement_ptr A,
    const size_t lda,
    const Givaro::FloatDomain::ConstElement_ptr X,
    const size_t incX,
    const Givaro::FloatDomain::Element beta,
    Givaro::FloatDomain::Element_ptr Y,
    const size_t incY,
    MMHelper< Givaro::FloatDomain, MMHelperAlgo::Classic, ModeCategories::DefaultTag
> & H) [inline]
```

**11.1.3.73 fgemv()** [10/19]

```
template<class Field , class Cut , class Param >
Field::Element_ptr fgemv (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const size_t m,
    const size_t n,
    const typename Field::Element alpha,
    const typename Field::ConstElement_ptr A,
    const size_t lda,
    const typename Field::ConstElement_ptr X,
    const size_t incX,
    const typename Field::Element beta,
    typename Field::Element_ptr Y,
    const size_t incY,
    ParSeqHelper::Parallel< Cut, Param > & parH)
```

**11.1.3.74 fgemv()** [11/19]

```
template<class Field >
Field::Element_ptr fgemv (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const size_t m,
    const size_t n,
    const typename Field::Element alpha,
    const typename Field::ConstElement_ptr A,
    const size_t lda,
    const typename Field::ConstElement_ptr X,
    const size_t incX,
    const typename Field::Element beta,
    typename Field::Element_ptr Y,
    const size_t incY,
    ParSeqHelper::Sequential & seqH)
```

**11.1.3.75 fgemv()** [12/19]

```
FFPACK::rns_double::Element_ptr fgemv (
    const FFPACK::RNSInteger< FFPACK::rns_double > & F,
    const FFLAS_TRANSPOSE ta,
    const size_t M,
    const size_t N,
    const FFPACK::rns_double::Element alpha,
    FFPACK::rns_double::ConstElement_ptr A,
    const size_t lda,
    FFPACK::rns_double::ConstElement_ptr X,
    const size_t incX,
    const FFPACK::rns_double::Element beta,
    FFPACK::rns_double::Element_ptr Y,
    const size_t incY,
    MMHelper< FFPACK::RNSInteger< FFPACK::rns_double >, MMHelperAlgo::Classic, ModeCategories::Default > & H) [inline]
```

**11.1.3.76 fgemv()** [13/19]

```
FFPACK::rns_double::Element_ptr fgemv (
    const FFPACK::RNSIntegerMod< FFPACK::rns_double > & F,
    const FFLAS_TRANSPOSE ta,
    const size_t M,
    const size_t N,
    const FFPACK::rns_double::Element alpha,
    FFPACK::rns_double::ConstElement_ptr A,
    const size_t lda,
    FFPACK::rns_double::ConstElement_ptr X,
    const size_t incX,
    const FFPACK::rns_double::Element beta,
    FFPACK::rns_double::Element_ptr Y,
    const size_t incY,
    MMHelper< FFPACK::RNSIntegerMod< FFPACK::rns_double >, MMHelperAlgo::Classic,
    ModeCategories::DefaultTag > & H) [inline]
```

**11.1.3.77 fgemv()** [14/19]

```
Givaro::Integer * fgemv (
    const Givaro::ZRing< Givaro::Integer > & F,
    const FFLAS_TRANSPOSE ta,
    const size_t m,
    const size_t n,
    const Givaro::Integer alpha,
    Givaro::Integer * A,
    const size_t lda,
    Givaro::Integer * X,
    const size_t ldx,
    Givaro::Integer beta,
    Givaro::Integer * Y,
    const size_t ldy,
    MMHelper< Givaro::ZRing< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo<
    ElementCategories::RNSElementTag > > & H) [inline]
```

**11.1.3.78 fgemv()** [15/19]

```
Givaro::Integer * fgemv (
    const Givaro::Modular< Givaro::Integer > & F,
    const FFLAS_TRANSPOSE ta,
    const size_t m,
    const size_t n,
    const Givaro::Integer alpha,
    Givaro::Integer * A,
    const size_t lda,
    Givaro::Integer * X,
    const size_t ldx,
    Givaro::Integer beta,
    Givaro::Integer * Y,
    const size_t ldy,
    MMHelper< Givaro::Modular< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo<
    ElementCategories::RNSElementTag > > & H) [inline]
```

**11.1.3.79 fgemv()** [16/19]

```

template<size_t K1, size_t K2, class ParSeq >
RecInt::ruint< K1 > * fgemv (
    const Givaro::Modular< RecInt::ruint< K1 >, RecInt::ruint< K2 > > & F,
    const FFLAS_TRANSPOSE ta,
    const size_t m,
    const size_t n,
    const RecInt::ruint< K1 > alpha,
    const RecInt::ruint< K1 > * A,
    const size_t lda,
    const RecInt::ruint< K1 > * X,
    const size_t incx,
    RecInt::ruint< K1 > beta,
    RecInt::ruint< K1 > * Y,
    const size_t incy,
    MMHelper< Givaro::Modular< RecInt::ruint< K1 >, RecInt::ruint< K2 > >, MMHelperAlgo::Classic,
    ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq > & H) [inline]

```

**11.1.3.80 fger()** [1/12]

```

template<class Field >
void fger (
    const Field & F,
    const size_t M,
    const size_t N,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr x,
    const size_t incx,
    typename Field::ConstElement_ptr y,
    const size_t incy,
    typename Field::Element_ptr A,
    const size_t lda) [inline]

```

fger: rank one update of a general matrix

Computes  $A \leftarrow \alpha x y^T + A$

**Parameters**

	$F$	field
	$M$	rows
	$N$	cols
	$\alpha$	scalar
in, out	$A$	dense matrix of size MxN and leading dimension lda
	$lda$	leading dimension of A
	$x$	dense vector of size M
	$incx$	stride of X
	$y$	dense vector of size N
	$incy$	stride of Y

**11.1.3.81 fger() [2/12]**

```

template<class Field >
void fger (
    const Field & F,
    const size_t M,
    const size_t N,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr x,
    const size_t incx,
    typename Field::ConstElement_ptr y,
    const size_t incy,
    typename Field::Element_ptr A,
    const size_t lda,
    MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::MachineFlo
> > & H) [inline]

```

**11.1.3.82 fger() [3/12]**

```

template<class Field , class AnyTag >
void fger (
    const Field & F,
    const size_t M,
    const size_t N,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr x,
    const size_t incx,
    typename Field::ConstElement_ptr y,
    const size_t incy,
    typename Field::Element_ptr A,
    const size_t lda,
    MMHelper< Field, MMHelperAlgo::Classic, AnyTag > & H) [inline]

```

**11.1.3.83 fger() [4/12]**

```

void fger (
    const Givaro::DoubleDomain & F,
    const size_t M,
    const size_t N,
    const Givaro::DoubleDomain::Element alpha,
    const Givaro::DoubleDomain::ConstElement_ptr x,
    const size_t incx,
    const Givaro::DoubleDomain::ConstElement_ptr y,
    const size_t incy,
    Givaro::DoubleDomain::Element_ptr A,
    const size_t lda,
    MMHelper< Givaro::DoubleDomain, MMHelperAlgo::Classic, ModeCategories::DefaultTag
> & H) [inline]

```

**11.1.3.84 fger()** [5/12]

```

template<class Field >
void fger (
    const Field & F,
    const size_t M,
    const size_t N,
    const typename Field::Element alpha,
    const typename Field::ConstElement_ptr x,
    const size_t incx,
    const typename Field::ConstElement_ptr y,
    const size_t incy,
    typename Field::Element_ptr A,
    const size_t lda,
    MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DefaultBoundedTag > & H)
[inline]

```

**11.1.3.85 fger()** [6/12]

```

void fger (
    const Givaro::FloatDomain & F,
    const size_t M,
    const size_t N,
    const Givaro::FloatDomain::Element alpha,
    const Givaro::FloatDomain::ConstElement_ptr x,
    const size_t incx,
    const Givaro::FloatDomain::ConstElement_ptr y,
    const size_t incy,
    Givaro::FloatDomain::Element_ptr A,
    const size_t lda,
    MMHelper< Givaro::FloatDomain, MMHelperAlgo::Classic, ModeCategories::DefaultTag
> & H) [inline]

```

**11.1.3.86 fger()** [7/12]

```

template<class Field >
void fger (
    const Field & F,
    const size_t M,
    const size_t N,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr x,
    const size_t incx,
    typename Field::ConstElement_ptr y,
    const size_t incy,
    typename Field::Element_ptr A,
    const size_t lda,
    MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::LazyTag > & H) [inline]

```

**11.1.3.87 fger()** [8/12]

```

template<class Field >
void fger (
    const Field & F,
    const size_t M,
    const size_t N,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr x,
    const size_t incx,
    typename Field::ConstElement_ptr y,
    const size_t incy,
    typename Field::Element_ptr A,
    const size_t lda,
    MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DelayedTag > & H) [inline]

```

**11.1.3.88 fger()** [9/12]

```

void fger (
    const Givaro::Modular< Givaro::Integer > & F,
    const size_t M,
    const size_t N,
    const typename Givaro::Integer alpha,
    typename Givaro::Integer * x,
    const size_t incx,
    typename Givaro::Integer * y,
    const size_t incy,
    typename Givaro::Integer * A,
    const size_t lda,
    MMHelper< Givaro::Modular< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo<
ElementCategories::RNSElementTag > > & H) [inline]

```

**11.1.3.89 fger()** [10/12]

```

template<typename RNS >
void fger (
    const FFPACK::RNSInteger< RNS > & F,
    const size_t M,
    const size_t N,
    const typename FFPACK::RNSInteger< RNS >::Element alpha,
    typename FFPACK::RNSInteger< RNS >::Element_ptr x,
    const size_t incx,
    typename FFPACK::RNSInteger< RNS >::Element_ptr y,
    const size_t incy,
    typename FFPACK::RNSInteger< RNS >::Element_ptr A,
    const size_t lda,
    MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag
> & H) [inline]

```



**11.1.3.90 fger()** [11/12]

```

template<typename RNS >
void fger (
    const FFPACK::RNSIntegerMod< RNS > & F,
    const size_t M,
    const size_t N,
    const typename FFPACK::RNSIntegerMod< RNS >::Element alpha,
    typename FFPACK::RNSIntegerMod< RNS >::Element_ptr x,
    const size_t incx,
    typename FFPACK::RNSIntegerMod< RNS >::Element_ptr y,
    const size_t incy,
    typename FFPACK::RNSIntegerMod< RNS >::Element_ptr A,
    const size_t lda,
    MMHelper< FFPACK::RNSIntegerMod< RNS >, MMHelperAlgo::Classic > & H) [inline]

```

**11.1.3.91 freduce()** [1/11]

```

template<class Field >
void freduce (
    const Field & F,
    const size_t n,
    typename Field::ConstElement_ptr Y,
    const size_t incY,
    typename Field::Element_ptr X,
    const size_t incX)

```

$\text{freduce } x \leftarrow y \bmod F.$

**Parameters**

$F$	field
$n$	size of the vectors
$Y$	vector of Element
$incY$	stride of Y
$X$	vector in F
$incX$	stride of X

**Bug** use `cblas_(d)scal` when possible

**11.1.3.92 freduce()** [2/11]

```

template<class Field >
void freduce (
    const Field & F,
    const size_t n,
    typename Field::Element_ptr X,
    const size_t incX)

```

$\text{freduce } x \leftarrow x \bmod F.$

## Parameters

$F$	field
$n$	size of the vectors
$X$	vector in $F$
$incX$	stride of $X$

**Bug** use `cblas_(d)scal` when possible

### 11.1.3.93 `freduce_constoverride()` [1/2]

```
template<class Field >
void freduce_constoverride (
    const Field & F,
    const size_t m,
    typename Field::ConstElement_ptr A,
    const size_t incX)
```

### 11.1.3.94 `finit()` [1/8]

```
template<class Field , class ConstOtherElement_ptr >
void finit (
    const Field & F,
    const size_t n,
    ConstOtherElement_ptr Y,
    const size_t incY,
    typename Field::Element_ptr X,
    const size_t incX)
```

### 11.1.3.95 `finit()` [2/8]

```
template<class Field >
void finit (
    const Field & F,
    const size_t n,
    typename Field::Element_ptr X,
    const size_t incX)
```

`finit` Initializes  $X$  in  $F$ .

## Parameters

$F$	field
$n$	size of the vectors
$X$	vector in $F$
$incX$	stride of $X$

**11.1.3.96 freduce()** [3/11]

```
template<class Field >
void freduce (
    const Field & F,
    const size_t m,
    const size_t n,
    typename Field::Element_ptr A,
    const size_t lda)
```

freduce  $A \leftarrow A \bmod F$ .

**Parameters**

$F$	field
$m$	number of rows
$n$	number of cols
$A$	matrix in F
$lda$	stride of A

**11.1.3.97 freduce()** [4/11]

```
template<class Field >
void freduce (
    const Field & F,
    const FFLAS_UPLO UpLo,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda)
```

freduce for square symmetric matrices

**11.1.3.98 pfreduce()**

```
template<class Field >
void pfreduce (
    const Field & F,
    const size_t m,
    const size_t n,
    typename Field::Element_ptr A,
    const size_t lda,
    const size_t numths)
```

**11.1.3.99 freduce()** [5/11]

```
template<class Field >
void freduce (
    const Field & F,
    const size_t m,
    const size_t n,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    typename Field::Element_ptr A,
    const size_t lda)
```

freduce  $A \leftarrow B \bmod F$ .

## Parameters

$F$	field
$m$	number of rows
$n$	number of cols
$A$	matrix in $F$
$lda$	stride of $A$
$B$	matrix in <code>Element</code>
$ldb$	stride of $B$

**11.1.3.100 `freduce_constoverride()` [2/2]**

```
template<class Field >
void freduce_constoverride (
    const Field & F,
    const size_t m,
    const size_t n,
    typename Field::ConstElement_ptr A,
    const size_t lda)
```

**11.1.3.101 `finit()` [3/8]**

```
template<class Field , class OtherElement_ptr >
void finit (
    const Field & F,
    const size_t m,
    const size_t n,
    const OtherElement_ptr B,
    const size_t ldb,
    typename Field::Element_ptr A,
    const size_t lda)
```

$\text{finit } A \leftarrow B \bmod F.$

## Parameters

$F$	field
$m$	number of rows
$n$	number of cols
$A$	matrix in $F$
$lda$	stride of $A$
$B$	matrix in <code>OtherElement</code>
$ldb$	stride of $B$

**11.1.3.102 `finit()` [4/8]**

```
template<class Field >
void finit (
    const Field & F,
    const size_t m,
    const size_t n,
    typename Field::Element_ptr A,
    const size_t lda)
```

**11.1.3.103 freduce()** [6/11]

```
template<>
void freduce (
    const FFPACK::RNSIntegerMod< FFPACK::rns_double > & F,
    const size_t n,
    FFPACK::RNSIntegerMod< FFPACK::rns_double >::Element_ptr A,
    size_t inc) [inline]
```

**11.1.3.104 freduce()** [7/11]

```
template<>
void freduce (
    const FFPACK::RNSIntegerMod< FFPACK::rns_double > & F,
    const size_t m,
    const size_t n,
    FFPACK::rns_double::Element_ptr A,
    size_t lda) [inline]
```

**11.1.3.105 freivalds()**

```
template<class Field >
bool freivalds (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    typename Field::ConstElement_ptr C,
    const size_t ldc) [inline]
```

freivalds: **F**reivalds **G**eneral **M**atrix **M**ultiply **R**andom **C**heck.

Randomly Checks  $C = \alpha \text{op}(A) \times \text{op}(B)$

**Parameters**

<i>F</i>	field.
<i>ta</i>	if ta==FflasTrans then $\text{op}(A) = A^t$ , else $\text{op}(A) = A$ ,
<i>tb</i>	same for matrix B
<i>m</i>	see A
<i>n</i>	see B
<i>k</i>	see A
<i>alpha</i>	scalar
<i>A</i>	$\text{op}(A)$ is $m \times k$
<i>B</i>	$\text{op}(B)$ is $k \times n$
<i>C</i>	$C$ is $m \times n$
<i>lda</i>	leading dimension of A
<i>ldb</i>	leading dimension of B
<i>ldc</i>	leading dimension of C

**11.1.3.106 fscaln()** [1/10]

```
template<class Field >
void fscaln (
    const Field & F,
    const size_t n,
    const typename Field::Element alpha,
    typename Field::Element_ptr X,
    const size_t incX) [inline]
```

$\text{fscaln } x \leftarrow \alpha \cdot x.$

**Parameters**

$F$	field
$n$	size of the vectors
$\alpha$	scalar
$X$	vector in $\mathbb{F}$
$\text{incX}$	stride of $X$

**Bug** use `cblas_(d)scal` when possible

**Todo** check if comparison with  $\pm 1, 0$  is necessary.

**11.1.3.107 fscal()** [1/10]

```
template<class Field >
void fscal (
    const Field & F,
    const size_t n,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr X,
    const size_t incX,
    typename Field::Element_ptr Y,
    const size_t incY) [inline]
```

$\text{fscal } y \leftarrow \alpha \cdot x.$

**Parameters**

	$F$	field
	$n$	size of the vectors
	$\alpha$	scalar
in	$X$	vector in $\mathbb{F}$
	$\text{incX}$	stride of $X$
out	$Y$	vector in $\mathbb{F}$
	$\text{incY}$	stride of $Y$

**Bug** use `cblas_(d)scal` when possible

**Todo** check if comparison with  $\pm 1, 0$  is necessary.

**11.1.3.108 fscal()** [2/10]

```
template<>
void fscal (
    const Givaro::DoubleDomain & ,
    const size_t N,
    const Givaro::DoubleDomain::Element a,
    Givaro::DoubleDomain::ConstElement_ptr x,
    const size_t incx,
    Givaro::DoubleDomain::Element_ptr y,
    const size_t incy) [inline]
```

**11.1.3.109 fscal()** [3/10]

```
template<>
void fscal (
    const Givaro::FloatDomain & ,
    const size_t N,
    const Givaro::FloatDomain::Element a,
    Givaro::FloatDomain::ConstElement_ptr x,
    const size_t incx,
    Givaro::FloatDomain::Element_ptr y,
    const size_t incy) [inline]
```

**11.1.3.110 fscaln()** [2/10]

```
template<>
void fscaln (
    const Givaro::DoubleDomain & ,
    const size_t N,
    const Givaro::DoubleDomain::Element a,
    Givaro::DoubleDomain::Element_ptr y,
    const size_t incy) [inline]
```

**11.1.3.111 fscaln()** [3/10]

```
template<>
void fscaln (
    const Givaro::FloatDomain & ,
    const size_t N,
    const Givaro::FloatDomain::Element a,
    Givaro::FloatDomain::Element_ptr y,
    const size_t incy) [inline]
```

**11.1.3.112 fscaln()** [4/10]

```
template<class Field >
void fscaln (
    const Field & F,
    const size_t m,
    const size_t n,
    const typename Field::Element alpha,
    typename Field::Element_ptr A,
    const size_t lda) [inline]
```

$\text{fscaln } A \leftarrow a \cdot A.$

## Parameters

$F$	field
$m$	number of rows
$n$	number of cols
$\alpha$	homotecie scalar
$A$	matrix in $F$
$lda$	stride of $A$

11.1.3.113 `fscal()` [4/10]

```
template<class Field >
void fscal (
    const Field & F,
    const size_t m,
    const size_t n,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::Element_ptr B,
    const size_t ldb) [inline]
```

$\text{fscal } B \leftarrow a \cdot A.$

## Parameters

	$F$	field
	$m$	number of rows
	$n$	number of cols
	$\alpha$	homotecie scalar
in	$A$	matrix in $F$
	$lda$	stride of $A$
out	$B$	matrix in $F$
	$ldb$	stride of $B$

11.1.3.114 `fscaln()` [5/10]

```
template<>
void fscaln (
    const FFPACK::RNSInteger< FFPACK::rns_double > & F,
    const size_t n,
    const FFPACK::rns_double::Element alpha,
    FFPACK::rns_double::Element_ptr A,
    const size_t inc) [inline]
```



**11.1.3.115 fscal()** [5/10]

```
template<>
void fscal (
    const FFPACK::RNSInteger< FFPACK::rns_double > & F,
    const size_t n,
    const FFPACK::rns_double::Element alpha,
    FFPACK::rns_double::ConstElement_ptr A,
    const size_t Ainc,
    FFPACK::rns_double::Element_ptr B,
    const size_t Binc) [inline]
```

**11.1.3.116 fscaln()** [6/10]

```
template<>
void fscaln (
    const FFPACK::RNSInteger< FFPACK::rns_double > & F,
    const size_t m,
    const size_t n,
    const FFPACK::rns_double::Element alpha,
    FFPACK::rns_double::Element_ptr A,
    const size_t lda) [inline]
```

**11.1.3.117 fscal()** [6/10]

```
template<>
void fscal (
    const FFPACK::RNSInteger< FFPACK::rns_double > & F,
    const size_t m,
    const size_t n,
    const FFPACK::rns_double::Element alpha,
    FFPACK::rns_double::ConstElement_ptr A,
    const size_t lda,
    FFPACK::rns_double::Element_ptr B,
    const size_t ldb) [inline]
```

**11.1.3.118 fscaln()** [7/10]

```
template<>
void fscaln (
    const FFPACK::RNSIntegerMod< FFPACK::rns_double > & F,
    const size_t n,
    const typename FFPACK::RNSIntegerMod< FFPACK::rns_double >::Element alpha,
    typename FFPACK::RNSIntegerMod< FFPACK::rns_double >::Element_ptr A,
    const size_t inc) [inline]
```

**11.1.3.119 fscal()** [7/10]

```
template<>
void fscal (
    const FFPACK::RNSIntegerMod< FFPACK::rns_double > & F,
    const size_t n,
    const FFPACK::rns_double::Element alpha,
    FFPACK::rns_double::ConstElement_ptr A,
    const size_t Ainc,
    FFPACK::rns_double::Element_ptr B,
    const size_t Binc) [inline]
```

**11.1.3.120 fscaln()** [8/10]

```
template<>
void fscaln (
    const FFPACK::RNSIntegerMod< FFPACK::rns_double > & F,
    const size_t m,
    const size_t n,
    const FFPACK::rns_double::Element alpha,
    FFPACK::rns_double::Element_ptr A,
    const size_t lda) [inline]
```

**11.1.3.121 fscal()** [8/10]

```
template<>
void fscal (
    const FFPACK::RNSIntegerMod< FFPACK::rns_double > & F,
    const size_t m,
    const size_t n,
    const FFPACK::rns_double::Element alpha,
    FFPACK::rns_double::ConstElement_ptr A,
    const size_t lda,
    FFPACK::rns_double::Element_ptr B,
    const size_t ldb) [inline]
```

**11.1.3.122 fsyr2k()**

```
template<class Field >
Field::Element_ptr fsyr2k (
    const Field & F,
    const FFLAS_UPLO UpLo,
    const FFLAS_TRANSPOSE trans,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc) [inline]
```

fsyr2k: Symmetric Rank 2K update

Computes the Lower or Upper triangular part of  $C = \alpha(A \times B^T + B \times A^T) + \beta C$  or  $C = \alpha(A^T \times B + B^T \times A) + \beta C$

## Parameters

<i>F</i>	field.
<i>UpLo</i>	whether to compute the upper or the lower triangular part of the symmetric matrix C
<i>trans</i>	if <code>ta==FflasNoTrans</code> then compute $C = \alpha(A \times B^T + B \times A^T) + \beta C$ , else $C = \alpha(A^T \times B + B^T \times A) + \beta C$
<i>n</i>	order of matrix C
<i>k</i>	see A
<i>alpha</i>	scalar
<i>A</i>	A is $n \times k$ (FflasNoTrans) or A is $k \times n$ (FflasTrans)
<i>lda</i>	leading dimension of A
<i>beta</i>	scalar
<i>C</i>	C is $n \times n$
<i>ldc</i>	leading dimension of C

## Warning

$\alpha$  must be invertible

## 11.1.3.123 fsyrk() [1/16]

```
template<class Field >
Field::Element_ptr fsyrk (
    const Field & F,
    const FFLAS_UPLO UpLo,
    const FFLAS_TRANSPOSE trans,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc) [inline]
```

fsyrk: Symmetric Rank K update

Computes the Lower or Upper triangular part of  $C = \alpha A \times A^T + \beta C$  or  $C = \alpha A^T \times A + \beta C$

## Parameters

<i>F</i>	field.
<i>UpLo</i>	whether to compute the upper or the lower triangular part of the symmetric matrix C
<i>trans</i>	if <code>ta==FflasNoTrans</code> then compute $C = \alpha A \times A^T + \beta C$ , else $C = \alpha A^T \times A + \beta C$
<i>n</i>	order of matrix C
<i>k</i>	see A
<i>alpha</i>	scalar
<i>A</i>	A is $n \times k$ or A is $k \times n$
<i>lda</i>	leading dimension of A
<i>beta</i>	scalar
<i>C</i>	C is $n \times n$
<i>ldc</i>	leading dimension of C

**Warning**

$\alpha$  *must* be invertible

**11.1.3.124 fsyrk()** [2/16]

```
template<class Field >
Field::Element_ptr fsyrk (
    const Field & F,
    const FFLAS_UPLO UpLo,
    const FFLAS_TRANSPOSE trans,
    const size_t N,
    const size_t K,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    const ParSeqHelper::Sequential seq) [inline]
```

**11.1.3.125 fsyrk()** [3/16]

```
template<class Field >
Field::Element_ptr fsyrk (
    const Field & F,
    const FFLAS_UPLO UpLo,
    const FFLAS_TRANSPOSE trans,
    const size_t N,
    const size_t K,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DefaultTag > & H) [inline]
```

**11.1.3.126 fsyrk()** [4/16]

```
template<class Field >
Field::Element_ptr fsyrk (
    const Field & F,
    const FFLAS_UPLO UpLo,
    const FFLAS_TRANSPOSE trans,
    const size_t N,
    const size_t K,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::MachineFlo
>, ParSeqHelper::Sequential > & H) [inline]
```

**11.1.3.127 fsyrk()** [5/16]

```

template<class Field >
Field::Element_ptr fsyrk (
    const Field & F,
    const FFLAS_UPLO UpLo,
    const FFLAS_TRANSPOSE trans,
    const size_t N,
    const size_t K,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DelayedTag > & H) [inline]

```

**11.1.3.128 fsyrk()** [6/16]

```

template<class Field >
Field::Element_ptr fsyrk (
    const Field & F,
    const FFLAS_UPLO UpLo,
    const FFLAS_TRANSPOSE trans,
    const size_t N,
    const size_t K,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::LazyTag > & H) [inline]

```

**11.1.3.129 fsyrk()** [7/16]

```

template<class Field , typename Mode >
Field::Element_ptr fsyrk (
    const Field & F,
    const FFLAS_UPLO UpLo,
    const FFLAS_TRANSPOSE trans,
    const size_t N,
    const size_t K,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::DivideAndConquer, Mode > & H) [inline]

```

**11.1.3.130 fsyrk() [8/16]**

```

template<class Field >
Field::Element_ptr fsyrk (
    const Field & F,
    const FFLAS_UPLO UpLo,
    const FFLAS_TRANSPOSE trans,
    const size_t N,
    const size_t K,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DefaultBoundedTag > & H)
[inline]

```

**11.1.3.131 fsyrk() [9/16]**

```

Givaro::FloatDomain::Element_ptr fsyrk (
    const Givaro::FloatDomain & F,
    const FFLAS_UPLO UpLo,
    const FFLAS_TRANSPOSE trans,
    const size_t N,
    const size_t K,
    const Givaro::FloatDomain::Element alpha,
    Givaro::FloatDomain::ConstElement_ptr A,
    const size_t lda,
    const Givaro::FloatDomain::Element beta,
    Givaro::FloatDomain::Element_ptr C,
    const size_t ldc,
    MMHelper< Givaro::FloatDomain, MMHelperAlgo::Classic, ModeCategories::DefaultTag
> & H) [inline]

```

**11.1.3.132 fsyrk() [10/16]**

```

Givaro::DoubleDomain::Element_ptr fsyrk (
    const Givaro::DoubleDomain & F,
    const FFLAS_UPLO UpLo,
    const FFLAS_TRANSPOSE trans,
    const size_t N,
    const size_t K,
    const Givaro::DoubleDomain::Element alpha,
    Givaro::DoubleDomain::ConstElement_ptr A,
    const size_t lda,
    const Givaro::DoubleDomain::Element beta,
    Givaro::DoubleDomain::Element_ptr C,
    const size_t ldc,
    MMHelper< Givaro::DoubleDomain, MMHelperAlgo::Classic, ModeCategories::DefaultTag
> & H) [inline]

```

**11.1.3.133 fsyrk()** [11/16]

```

template<class Field >
Field::Element_ptr fsyrk (
    const Field & F,
    const FFLAS_UPLO UpLo,
    const FFLAS_TRANSPOSE trans,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr D,
    const size_t incD,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    const size_t threshold = __FFLASFFPACK_FSYRK_THRESHOLD) [inline]

```

fsyrk: Symmetric Rank K update with diagonal scaling

Computes the Lower or Upper triangular part of  $C = \alpha A \times D \times A^T + \beta C$  or  $C = \alpha A^T \times D \times A + \beta C$  where D is a diagonal matrix. Matrix A is updated into  $D \times A$  (if trans = FflasTrans) or  $A \times D$  (if trans = FflasNoTrans).

**Parameters**

<i>F</i>	field.
<i>UpLo</i>	whether to compute the upper or the lower triangular part of the symmetric matrix C
<i>trans</i>	if ta==FflasNoTrans then compute $C = \alpha A \times A^T + \beta C$ , else $C = \alpha A^T \times A + \beta C$
<i>n</i>	order of matrix C
<i>k</i>	see A
<i>alpha</i>	scalar
<i>A</i>	A is $n \times k$ or A is $k \times n$
<i>lda</i>	leading dimension of A
<i>D</i>	D is $k \times k$ diagonal matrix, stored as a vector of k coefficients
<i>lda</i>	leading dimension of A
<i>beta</i>	scalar
<i>C</i>	C is $n \times n$
<i>ldc</i>	leading dimension of C

**Warning**

$\alpha$  must be invertible

**11.1.3.134 fsyrk()** [12/16]

```

template<class Field >
Field::Element_ptr fsyrk (
    const Field & F,
    const FFLAS_UPLO UpLo,
    const FFLAS_TRANSPOSE trans,
    const size_t N,

```

```

const size_t K,
const typename Field::Element alpha,
typename Field::Element_ptr A,
const size_t lda,
typename Field::ConstElement_ptr D,
const size_t incD,
const typename Field::Element beta,
typename Field::Element_ptr C,
const size_t ldc,
const ParSeqHelper::Sequential seq,
const size_t threshold) [inline]

```

#### 11.1.3.135 fsyrk() [13/16]

```

template<class Field , class Cut , class Param >
Field::Element_ptr fsyrk (
    const Field & F,
    const FFLAS_UPLO UpLo,
    const FFLAS_TRANSPOSE trans,
    const size_t N,
    const size_t K,
    const typename Field::Element alpha,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr D,
    const size_t incD,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    const ParSeqHelper::Parallel< Cut, Param > par,
    const size_t threshold) [inline]

```

#### 11.1.3.136 fsyrk() [14/16]

```

template<class Field >
Field::Element_ptr fsyrk (
    const Field & F,
    const FFLAS_UPLO UpLo,
    const FFLAS_TRANSPOSE trans,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr D,
    const size_t incD,
    const std::vector< bool > & twoBlock,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    const size_t threshold = __FFLASFFPACK_FSYRK_THRESHOLD) [inline]

```

fsyrk: Symmetric Rank K update with diagonal scaling

Computes the Lower or Upper triangular part of  $C = \alpha A \times \text{Delta} D \times A^T + \beta C$  or  $C = \alpha A^T \times \text{Delta} D \times A + \beta C$  where D is a diagonal matrix and Delta is a block diagonal with either 1 on the diagonal or 2x2 swap blocks Matrix A is updated into  $D \times A$  (if trans = FflasTrans) or  $A \times D$  (if trans = FflasNoTrans).



## Parameters

<i>F</i>	field.
<i>UpLo</i>	whether to compute the upper or the lower triangular part of the symmetric matrix <i>C</i>
<i>trans</i>	if <code>ta==FflasNoTrans</code> then compute $C = \alpha A \Delta D \times A^T + \beta C$ , else $C = \alpha A^T \Delta D \times A + \beta C$
<i>n</i>	see <i>B</i>
<i>k</i>	see <i>A</i>
<i>alpha</i>	scalar
<i>A</i>	<i>A</i> is $n \times k$ or <i>A</i> is $k \times n$
<i>lda</i>	leading dimension of <i>A</i>
<i>D</i>	<i>D</i> is $k \times k$ diagonal matrix, stored as a vector of <i>k</i> coefficients
<i>twoBlocks</i>	a vector boolean indicating the beginning of each 2x2 blocs in <i>Delta</i>
<i>lda</i>	leading dimension of <i>A</i>
<i>beta</i>	scalar
<i>C</i>	<i>C</i> is $n \times n$
<i>ldc</i>	leading dimension of <i>C</i>

## Warning

$\alpha$  must be invertible

## 11.1.3.137 computeS1S2()

```
template<class Field , class FieldTrait >
void computeS1S2 (
    const Field & F,
    const FFLAS_TRANSPOSE trans,
    const size_t N,
    const size_t K,
    const typename Field::Element x,
    const typename Field::Element y,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr S,
    const size_t lds,
    typename Field::Element_ptr T,
    const size_t ldt,
    MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]
```

## 11.1.3.138 fsyrk() [15/16]

```
template<class Field >
Field::Element_ptr fsyrk (
    const Field & F,
    const FFLAS_UPLO UpLo,
    const FFLAS_TRANSPOSE trans,
    const size_t N,
    const size_t K,
    const typename Field::Element alpha,
```

```

    typename Field::ConstElement_ptr A,
    const size_t lda,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::Winograd, ModeCategories::DelayedTag, ParSeqHelper::Sequential
> & H) [inline]

```

#### 11.1.3.139 fsyrk() [16/16]

```

template<class Field , class Mode >
Field::Element_ptr fsyrk (
    const Field & F,
    const FFLAS_UPLO UpLo,
    const FFLAS_TRANSPOSE trans,
    const size_t N,
    const size_t K,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::Winograd, Mode > & H) [inline]

```

#### 11.1.3.140 fsyrk\_strassen() [1/2]

```

template<class Field , class FieldTrait >
Field::Element_ptr fsyrk_strassen (
    const Field & F,
    const FFLAS_UPLO uplo,
    const FFLAS_TRANSPOSE trans,
    const size_t N,
    const size_t K,
    const typename Field::Element y1,
    const typename Field::Element y2,
    const typename Field::Element alpha,
    typename Field::Element_ptr A,
    const size_t lda,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]

```

#### 11.1.3.141 ftrmm() [1/3]

```

template<class Field >
void ftrmm (
    const Field & F,
    const FFLAS_SIDE Side,
    const FFLAS_UPLO UpLo,
    const FFLAS_TRANSPOSE TransA,
    const FFLAS_DIAG Diag,

```

```

const size_t M,
const size_t N,
const typename Field::Element alpha,
typename Field::ConstElement_ptr A,
const size_t lda,
typename Field::Element_ptr B,
const size_t ldb) [inline]

```

ftmmm: **TR**angular **M**atrix **M**ultiply.

Computes  $B \leftarrow \alpha \text{op}(A)B$  or  $B \leftarrow \alpha B \text{op}(A)$ .

#### Parameters

<i>F</i>	field
<i>Side</i>	if Side==FflasLeft then $B \leftarrow \alpha \text{op}(A)B$ is computed.
<i>Uplo</i>	if Uplo==FflasUpper then A is upper triangular
<i>TransA</i>	if TransA==FflasTrans then $\text{op}(A) = A^t$ .
<i>Diag</i>	if Diag==FflasUnit then A is implicitly unit.
<i>M</i>	rows of B
<i>N</i>	cols of B
<i>alpha</i>	scalar
<i>A</i>	triangular matrix. If Side==FflasLeft then A is $N \times N$ , otherwise A is $M \times M$
<i>lda</i>	leading dim of A
<i>B</i>	matrix of size MxN
<i>ldb</i>	leading dim of B

#### 11.1.3.142 ftmmm() [2/3]

```

template<class Field >
void ftmmm (
    const Field & F,
    const FFLAS_SIDE Side,
    const FFLAS_UPLO Uplo,
    const FFLAS_TRANSPOSE TransA,
    const FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc) [inline]

```

ftmmm: **TR**angular **M**atrix **M**ultiply with 3 operands Computes  $C \leftarrow \alpha \text{op}(A)B + \beta C$  or  $C \leftarrow \alpha B \text{op}(A) + \beta C$ .

#### Parameters

<i>F</i>	field
<i>Side</i>	if Side==FflasLeft then $B \leftarrow \alpha \text{op}(A)B$ is computed.

## Parameters

<i>Uplo</i>	if <code>Uplo==FflasUpper</code> then A is upper triangular
<i>TransA</i>	if <code>TransA==FflasTrans</code> then $\text{op}(A) = A^t$ .
<i>Diag</i>	if <code>Diag==FflasUnit</code> then A is implicitly unit.
<i>M</i>	rows of B
<i>N</i>	cols of B
<i>alpha</i>	scalar
<i>A</i>	triangular matrix. If <code>Side==FflasLeft</code> then A is $N \times N$ , otherwise A is $M \times M$
<i>lda</i>	leading dim of A
<i>B</i>	matrix of size $M \times N$
<i>ldb</i>	leading dim of B
<i>beta</i>	scalar
<i>C</i>	matrix of size $M \times N$
<i>ldc</i>	leading dim of C

11.1.3.143 `ftrsm()` [1/9]

```
template<class Field >
void ftrsm (
    const Field & F,
    const FFLAS_SIDE Side,
    const FFLAS_UPLO Uplo,
    const FFLAS_TRANSPOSE TransA,
    const FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    const typename Field::Element alpha,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr B,
    const size_t ldb) [inline]
```

11.1.3.144 `ftrsm()` [2/9]

```
template<class Field >
void ftrsm (
    const Field & F,
    const FFLAS_SIDE Side,
    const FFLAS_UPLO Uplo,
    const FFLAS_TRANSPOSE TransA,
    const FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    const typename Field::Element alpha,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr B,
    const size_t ldb,
    const ParSeqHelper::Sequential & PSH) [inline]
```

**11.1.3.145 ftrsm()** [3/9]

```
template<class Field , class Cut , class Param >
void ftrsm (
    const Field & F,
    const FFLAS_SIDE Side,
    const FFLAS_UPLO Uplo,
    const FFLAS_TRANSPOSE TransA,
    const FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    const typename Field::Element alpha,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr B,
    const size_t ldb,
    const ParSeqHelper::Parallel< Cut, Param > & PSH) [inline]
```

**11.1.3.146 ftrsm()** [4/9]

```
template<class Field , class ParSeqTrait = ParSeqHelper::Sequential>
void ftrsm (
    const Field & F,
    const FFLAS_SIDE Side,
    const FFLAS_UPLO Uplo,
    const FFLAS_TRANSPOSE TransA,
    const FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    const typename Field::Element alpha,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr B,
    const size_t ldb,
    TRSMHelper< StructureHelper::Recursive, ParSeqTrait > & H) [inline]
```

**11.1.3.147 ftrsm()** [5/9]

```
void ftrsm (
    const Givaro::Modular< Givaro::Integer > & F,
    const FFLAS_SIDE Side,
    const FFLAS_UPLO Uplo,
    const FFLAS_TRANSPOSE TransA,
    const FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    const Givaro::Integer alpha,
    const Givaro::Integer * A,
    const size_t lda,
    Givaro::Integer * B,
    const size_t ldb) [inline]
```

### 11.1.3.148 cblas\_impstrsm()

```
void cblas_impstrsm (
    const enum FFLAS_ORDER Order,
    const enum FFLAS_SIDE Side,
    const enum FFLAS_UPLO Uplo,
    const enum FFLAS_TRANSPOSE TransA,
    const enum FFLAS_DIAG Diag,
    const int M,
    const int N,
    const FFPACK::rns_double_elt alpha,
    FFPACK::rns_double_elt_cstptr A,
    const int lda,
    FFPACK::rns_double_elt_ptr B,
    const int ldb) [inline]
```

### 11.1.3.149 ftrsv() [1/2]

```
template<class Field >
void ftrsv (
    const Field & F,
    const FFLAS_UPLO Uplo,
    const FFLAS_TRANSPOSE TransA,
    const FFLAS_DIAG Diag,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::Element_ptr X,
    int incX) [inline]
```

ftrsv: TRIangular System solve with Vector Computes  $X \leftarrow \text{op}(A^{-1})X$

#### Parameters

<i>F</i>	field
<i>X</i>	vector of size N on a field F
<i>incX</i>	stride of X
<i>A</i>	a matrix of leading dimension lda and size N
<i>lda</i>	leading dimension of A
<i>N</i>	number of rows or columns of A according to TransA
<i>TransA</i>	if TransA==FflasTrans then $\text{op}(A) = A^t$ .
<i>Diag</i>	if Diag==FflasUnit then A is unit.
<i>Uplo</i>	if Uplo==FflasUpper then A is upper triangular

### 11.1.3.150 igemm\_()

```
void igemm_ (
    const enum FFLAS_ORDER Order,
    const enum FFLAS_TRANSPOSE TransA,
    const enum FFLAS_TRANSPOSE TransB,
    const size_t M,
```

```

    const size_t N,
    const size_t K,
    const int64_t alpha,
    const int64_t * A,
    const size_t lda,
    const int64_t * B,
    const size_t ldb,
    const int64_t beta,
    int64_t * C,
    const size_t ldc) [inline]

```

### 11.1.3.151 finit() [5/8]

```

template<class Field , class OtherElement_ptr >
void finit (
    const Field & F,
    const size_t n,
    const OtherElement_ptr Y,
    const size_t incY,
    typename Field::Element_ptr X,
    const size_t incX)

```

$\text{finit } x \leftarrow y \bmod F.$

#### Parameters

$F$	field
$n$	size of the vectors
$Y$	vector of OtherElement
$incY$	stride of Y
$X$	vector in F
$incX$	stride of X

**Bug** use cblas\_(d)scal when possible

### 11.1.3.152 fconvert() [1/3]

```

template<class Field , class OtherElement_ptr >
void fconvert (
    const Field & F,
    const size_t n,
    OtherElement_ptr X,
    const size_t incX,
    typename Field::ConstElement_ptr Y,
    const size_t incY)

```

$\text{fconvert } x \leftarrow y \bmod F.$

#### Parameters

$F$	field
$n$	size of the vectors

$Y$	vector of $F$
$incY$	stride of $Y$
$X$	vector in <code>OtherElement</code>
$incX$	stride of $X$

**Bug** use `cblas_(d)scal` when possible

#### 11.1.3.153 `fnegin()` [1/4]

```
template<class Field >
void fnegin (
    const Field & F,
    const size_t n,
    typename Field::Element_ptr X,
    const size_t incX)
```

`fnegin`  $x \leftarrow -x$ .

##### Parameters

$F$	field
$n$	size of the vectors
$X$	vector in $F$
$incX$	stride of $X$

**Bug** use `cblas_(d)scal` when possible

#### 11.1.3.154 `fneg()` [1/4]

```
template<class Field >
void fneg (
    const Field & F,
    const size_t n,
    typename Field::ConstElement_ptr Y,
    const size_t incY,
    typename Field::Element_ptr X,
    const size_t incX)
```

`fneg`  $x \leftarrow -y$ .

##### Parameters

$F$	field
$n$	size of the vectors
$X$	vector in $F$
$incX$	stride of $X$
$Y$	vector in $F$
$incY$	stride of $Y$

**Bug** use `cblas_(d)scal` when possible



**11.1.3.155 fzero()** [1/5]

```
template<class Field >
void fzero (
    const Field & F,
    const size_t n,
    typename Field::Element_ptr X,
    const size_t incX)
```

$\text{fzero} : A \leftarrow 0.$

**Parameters**

$F$	field
$n$	number of elements to zero
$X$	vector in $F$
$\text{incX}$	stride of $X$

**11.1.3.156 frand()** [1/2]

```
template<class Field , class RandIter >
void frand (
    const Field & F,
    RandIter & G,
    const size_t n,
    typename Field::Element_ptr X,
    const size_t incX)
```

$\text{frand} : A \leftarrow \text{random}.$

**Parameters**

$F$	field
$G$	randomiterator
$n$	number of elements to randomize
$X$	vector in $F$
$\text{incX}$	stride of $X$

**11.1.3.157 fiszero()** [1/4]

```
template<class Field >
bool fiszero (
    const Field & F,
    const size_t n,
    typename Field::ConstElement_ptr X,
    const size_t incX)
```

$\text{fiszero} : \text{test } X = 0.$

## Parameters

$F$	field
$n$	vector dimension
$X$	vector in $F$
$incX$	increment of $X$

**11.1.3.158 fequal()** [1/4]

```
template<class Field >
bool fequal (
    const Field & F,
    const size_t n,
    typename Field::ConstElement_ptr X,
    const size_t incX,
    typename Field::ConstElement_ptr Y,
    const size_t incY)
```

**fequal** : test  $X = Y$ .

## Parameters

$F$	field
$n$	vector dimension
$X$	vector in $F$
$incX$	increment of $X$
$Y$	vector in $F$
$incY$	increment of $Y$

**11.1.3.159 faxpby()** [1/2]

```
template<class Field >
void faxpby (
    const Field & F,
    const size_t N,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr X,
    const size_t incX,
    const typename Field::Element beta,
    typename Field::Element_ptr Y,
    const size_t incY)
```

**faxpby** :  $y \leftarrow \alpha \cdot x + \beta \cdot y$ .

## Parameters

	$F$	field
	$N$	size of the vectors
	$\alpha$	scalar
in	$X$	vector in $F$
	$incX$	stride of $X$

	<i>beta</i>	scalar
<i>in, out</i>	<i>Y</i>	vector in F
	<i>incY</i>	stride of Y

**Note**

this is a catlas function

**11.1.3.160 fdot()** [9/11]

```
template<typename Field , class Cut , class Param >
Field::Element fdot (
    const Field & F,
    const size_t N,
    typename Field::ConstElement_ptr X,
    const size_t incX,
    typename Field::ConstElement_ptr Y,
    const size_t incY,
    const ParSeqHelper::Parallel< Cut, Param > par) [inline]
```

**11.1.3.161 fswap()** [1/2]

```
template<class Field >
void fswap (
    const Field & F,
    const size_t N,
    typename Field::Element_ptr X,
    const size_t incX,
    typename Field::Element_ptr Y,
    const size_t incY)
```

fswap:  $X \leftrightarrow Y$ .

**Bug** use `cblas_dswap` when double

**Parameters**

<i>F</i>	field
<i>N</i>	size of the vectors
<i>X</i>	vector in F
<i>incX</i>	stride of X
<i>Y</i>	vector in F
<i>incY</i>	stride of Y

**11.1.3.162 fzero()** [2/5]

```
template<class Field >
void fzero (
    const Field & F,
    const size_t m,
    const size_t n,
    typename Field::Element_ptr A,
    const size_t lda)
```

$\text{fzero} : A \leftarrow 0.$

## Parameters

$F$	field
$m$	number of rows to zero
$n$	number of cols to zero
$A$	matrix in $F$
$lda$	stride of $A$

## Warning

may be buggy if Element is larger than int

**11.1.3.163 fzero()** [3/5]

```
template<class Field >
void fzero (
    const Field & F,
    const FFLAS_UPLO shape,
    const FFLAS_DIAG diag,
    const size_t n,
    typename Field::Element_ptr A,
    const size_t lda)
```

$fzero : A \leftarrow 0$  for a triangular matrix.

## Parameters

$F$	field
$shape$	shape of the triangular matrix
$m$	number of rows to zero
$n$	number of cols to zero
$A$	matrix in $F$
$lda$	stride of $A$

## Warning

may be buggy if Element is larger than int

**11.1.3.164 frand()** [2/2]

```
template<class Field , class RandIter >
void frand (
    const Field & F,
    RandIter & G,
    const size_t m,
    const size_t n,
    typename Field::Element_ptr A,
    const size_t lda)
```

$frand : A \leftarrow random.$

## Parameters

$F$	field
$G$	randomiterator
$m$	number of rows to randomize
$n$	number of cols to randomize
$A$	matrix in $F$
$lda$	stride of $A$

**11.1.3.165 fequal()** [2/4]

```
template<class Field >
bool fequal (
    const Field & F,
    const size_t m,
    const size_t n,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb)
```

fequal : test  $A = B$ .

## Parameters

$F$	field
$m$	row dimension
$n$	column dimension
$A$	$m \times n$ matrix in $F$
$lda$	leading dimension of $A$
$B$	$m \times n$ matrix in $F$
$ldb$	leading dimension of $B$

**11.1.3.166 fiszero()** [2/4]

```
template<class Field >
bool fiszero (
    const Field & F,
    const size_t m,
    const size_t n,
    typename Field::ConstElement_ptr A,
    const size_t lda)
```

fiszero : test  $A = 0$ .

## Parameters

$F$	field
$m$	row dimension
$n$	column dimension
$A$	$m \times n$ matrix in $F$
$lda$	leading dimension of $A$

**11.1.3.167 fidentity()** [1/4]

```
template<class Field >
void fidentity (
    const Field & F,
    const size_t m,
    const size_t n,
    typename Field::Element_ptr A,
    const size_t lda,
    const typename Field::Element & d)
```

creates a diagonal matrix

**11.1.3.168 fidentity()** [2/4]

```
template<class Field >
void fidentity (
    const Field & F,
    const size_t m,
    const size_t n,
    typename Field::Element_ptr A,
    const size_t lda)
```

creates a diagonal matrix

**11.1.3.169 finit()** [6/8]

```
template<class Field , class OtherElement_ptr >
void finit (
    const Field & F,
    const size_t m,
    const size_t n,
    typename Field::Element_ptr A,
    const size_t lda)
```

finit Initializes A in F\$.

**Parameters**

$F$	field
$m$	number of rows
$n$	number of cols
$A$	matrix in F
$lda$	stride of A

**11.1.3.170 fconvert()** [2/3]

```
template<class Field , class OtherElement_ptr >
void fconvert (
    const Field & F,
    const size_t m,
    const size_t n,
    OtherElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb)
```

fconvert  $A \leftarrow B \bmod F$ .

## Parameters

$F$	field
$m$	number of rows
$n$	number of cols
$A$	matrix in OtherElement
$lda$	stride of A
$B$	matrix in F
$ldb$	stride of B

**Todo** check if  $n == lda$

## 11.1.3.171 fnegin() [2/4]

```
template<class Field >
void fnegin (
    const Field & F,
    const size_t m,
    const size_t n,
    typename Field::Element_ptr A,
    const size_t lda)
```

$\text{fnegin } A \leftarrow -A.$

## Parameters

$F$	field
$m$	number of rows
$n$	number of cols
$A$	matrix in F
$lda$	stride of A

**Todo** check if  $n == lda$

## 11.1.3.172 fneg() [2/4]

```
template<class Field >
void fneg (
    const Field & F,
    const size_t m,
    const size_t n,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    typename Field::Element_ptr A,
    const size_t lda)
```

$\text{fneg } A \leftarrow -B.$



## Parameters

$F$	field
$m$	number of rows
$n$	number of cols
$A$	matrix in $F$
$lda$	stride of $A$

**Todo** check if  $n == lda$

**11.1.3.173 faxpby()** [2/2]

```
template<class Field >
void faxpby (
    const Field & F,
    const size_t m,
    const size_t n,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr X,
    const size_t ldx,
    const typename Field::Element beta,
    typename Field::Element_ptr Y,
    const size_t ldy)
```

$\text{faxpby} : y \leftarrow \alpha \cdot x + \beta \cdot y.$

## Parameters

	$F$	field
	$m$	row dimension
	$n$	column dimension
	$alpha$	scalar
in	$X$	vector in $F$
	$ldx$	leading dimension of $X$
	$beta$	scalar
in, out	$Y$	vector in $F$
	$ldy$	leading dimension of $Y$

## Note

this is a catlas function

**11.1.3.174 fmove()** [1/2]

```
template<class Field >
void fmove (
    const Field & F,
    const size_t m,
    const size_t n,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr B,
    const size_t ldb)
```

$\text{fmove} : A \leftarrow B \text{ and } B \leftarrow 0.$

## Parameters

$F$	field
$m$	number of rows to copy
$n$	number of cols to copy
$A$	matrix in $F$
$lda$	stride of $A$
$B$	matrix in $F$
$ldb$	stride of $B$

**11.1.3.175 bitsize()**

```
template<class Field >
size_t bitsize (
    const Field & F,
    size_t M,
    size_t N,
    const typename Field::ConstElement_ptr A,
    size_t lda) [inline]
```

bitsize: Computes the largest bitsize of the matrix' coefficients.

If the matrix is over a modular prime field, it returns the bitsize of the largest element (in a bsolute value)

## Parameters

$F$	field
$M$	rows
$N$	cols
$incX$	stride of $X$
$A$	a matrix of leading dimension $lda$ and size $M \times N$
$lda$	leading dimension of $A$

**11.1.3.176 bitsize< Givaro::ZRing< Givaro::Integer > >()**

```
template<>
size_t bitsize< Givaro::ZRing< Givaro::Integer > > (
    const Givaro::ZRing< Givaro::Integer > & F,
    size_t M,
    size_t N,
    const Givaro::Integer * A,
    size_t lda) [inline]
```

**11.1.3.177 ftrmv()**

```
template<class Field >
void ftrmv (
    const Field & F,
    const FFLAS_UPLO Uplo,
    const FFLAS_TRANSPOSE TransA,
    const FFLAS_DIAG Diag,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::Element_ptr X,
    int incX)
```

ftrsm: TRiangular Matrix Vector prodcut Computes  $X \leftarrow \text{op}(A)X$

**Parameters**

<i>F</i>	field
<i>X</i>	vector of size N on a field F
<i>incX</i>	stride of X
<i>A</i>	a matrix of leading dimension lda and size N
<i>lda</i>	leading dimension of A
<i>N</i>	number of rows and columns of A
<i>TransA</i>	if TransA==FflasTrans then $\text{op}(A) = A^T$ .
<i>Diag</i>	if Diag==FflasUnit then A is unit diagonal.
<i>Uplo</i>	if Uplo==FflasUpper then A is upper triangular

**11.1.3.178 ftrsm() [6/9]**

```
template<class Field >
void ftrsm (
    const Field & F,
    const FFLAS_SIDE Side,
    const FFLAS_UPLO Uplo,
    const FFLAS_TRANSPOSE TransA,
    const FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::Element_ptr B,
    const size_t ldb)
```

ftrsm: TRiangular System solve with Matrix.

Computes  $B \leftarrow \alpha \text{op}(A^{-1})B$  or  $B \leftarrow \alpha B \text{op}(A^{-1})$ .

**Parameters**

<i>F</i>	field
<i>Side</i>	if Side==FflasLeft then $B \leftarrow \alpha \text{op}(A^{-1})B$ is computed.

## Parameters

<i>Uplo</i>	if <code>Uplo==FflasUpper</code> then A is upper triangular
<i>TransA</i>	if <code>TransA==FflasTrans</code> then $\text{op}(A) = A^t$ .
<i>Diag</i>	if <code>Diag==FflasUnit</code> then A is unit.
<i>M</i>	rows of B
<i>N</i>	cols of B
<i>alpha</i>	scalar
<i>A</i>	triangular invertible matrix. If <code>Side==FflasLeft</code> then A is $N \times N$ , otherwise A is $M \times M$
<i>lda</i>	leading dim of A
<i>B</i>	matrix of size MxN
<i>ldb</i>	leading dim of B

**Bug**  $\alpha$  must be non zero.

## 11.1.3.179 fsyrk\_strassen() [2/2]

```
template<class Field , typename FieldTrait >
Field::Element_ptr fsyrk_strassen (
    const Field & F,
    const FFLAS_UPLO UpLo,
    const FFLAS_TRANSPOSE trans,
    const size_t N,
    const size_t K,
    const typename Field::Element y1,
    const typename Field::Element y2,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & H) [inline]
```

## 11.1.3.180 pfgemm() [1/7]

```
template<typename Field >
Field::Element_ptr pfgemm (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    size_t numthreads = 0)
```

**11.1.3.181 pfgemm\_1D\_rec()**

```

template<class Field >
Field::Element * pfgemm_1D_rec (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    const typename Field::Element_ptr A,
    const size_t lda,
    const typename Field::Element_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element * C,
    const size_t ldc,
    size_t seuil)

```

**11.1.3.182 pfgemm\_2D\_rec()**

```

template<class Field >
Field::Element * pfgemm_2D_rec (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    const typename Field::Element_ptr A,
    const size_t lda,
    const typename Field::Element_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element * C,
    const size_t ldc,
    size_t seuil)

```

**11.1.3.183 pfgemm\_3D\_rec()**

```

template<class Field >
Field::Element * pfgemm_3D_rec (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    const typename Field::Element_ptr A,
    const size_t lda,
    const typename Field::Element_ptr B,

```

```

    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    size_t seuil,
    size_t * x)

```

#### 11.1.3.184 pfgemm\_3D\_rec2()

```

template<class Field >
Field::Element_ptr pfgemm_3D_rec2 (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    const typename Field::Element_ptr A,
    const size_t lda,
    const typename Field::Element_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    size_t seuil,
    size_t * x)

```

#### 11.1.3.185 fgemm() [19/23]

```

template<class Field , class ModeTrait , class Strat , class Param >
std::enable_if<!std::is_same< ModeTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag
> >::value, typename Field::Element_ptr >::type fgemm (
    const Field & F,
    const FFLAS::FFLAS_TRANSPOSE ta,
    const FFLAS::FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::Winograd, ModeTrait, ParSeqHelper::Parallel< Strat,
Param > > & H) [inline]

```

#### 11.1.3.186 ftrsm() [7/9]

```

template<class Field , class Cut , class Param >
Field::Element_ptr ftrsm (

```

```

    const Field & F,
    const FFLAS::FFLAS_SIDE Side,
    const FFLAS::FFLAS_UPLO UpLo,
    const FFLAS::FFLAS_TRANSPOSE TA,
    const FFLAS::FFLAS_DIAG Diag,
    const size_t m,
    const size_t n,
    const typename Field::Element alpha,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr B,
    const size_t ldb,
    TRSMHelper< StructureHelper::Iterative, ParSeqHelper::Parallel< Cut, Param > > &
H) [inline]

```

#### 11.1.3.187 ftrsm() [8/9]

```

template<class Field , class Cut , class Param >
Field::Element_ptr ftrsm (
    const Field & F,
    const FFLAS::FFLAS_SIDE Side,
    const FFLAS::FFLAS_UPLO UpLo,
    const FFLAS::FFLAS_TRANSPOSE TA,
    const FFLAS::FFLAS_DIAG Diag,
    const size_t m,
    const size_t n,
    const typename Field::Element alpha,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr B,
    const size_t ldb,
    TRSMHelper< StructureHelper::Hybrid, ParSeqHelper::Parallel< Cut, Param > > & H)
[inline]

```

#### 11.1.3.188 fspmv() [1/2]

```

template<class Field , class SM >
void fspmv (
    const Field & F,
    const SM & A,
    typename Field::ConstElement_ptr x,
    const typename Field::Element & beta,
    typename Field::Element_ptr y) [inline]

```

#### 11.1.3.189 fspmm()

```

template<class Field , class SM >
void fspmm (
    const Field & F,
    const SM & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    const typename Field::Element & beta,
    typename Field::Element_ptr y,
    int ldy) [inline]

```

**11.1.3.190 sparse\_init() [1/16]**

```
template<class Field , class IndexT >
void sparse_init (
    const Field & F,
    Sparse< Field, SparseMatrix_t::COO > & A,
    const IndexT * row,
    const IndexT * col,
    typename Field::ConstElement_ptr dat,
    uint64_t rowdim,
    uint64_t coldim,
    uint64_t nnz) [inline]
```

**11.1.3.191 sparse\_init() [2/16]**

```
template<class Field , class IndexT >
void sparse_init (
    const Field & F,
    Sparse< Field, SparseMatrix_t::COO_ZO > & A,
    const IndexT * row,
    const IndexT * col,
    typename Field::ConstElement_ptr dat,
    uint64_t rowdim,
    uint64_t coldim,
    uint64_t nnz) [inline]
```

**11.1.3.192 sparse\_delete() [1/12]**

```
template<class Field >
void sparse_delete (
    const Sparse< Field, SparseMatrix_t::COO > & A) [inline]
```

**11.1.3.193 sparse\_delete() [2/12]**

```
template<class Field >
void sparse_delete (
    const Sparse< Field, SparseMatrix_t::COO_ZO > & A) [inline]
```

**11.1.3.194 sparse\_init() [3/16]**

```
template<class Field , class IndexT >
void sparse_init (
    const Field & F,
    Sparse< Field, SparseMatrix_t::CSR > & A,
    const IndexT * row,
    const IndexT * col,
    typename Field::ConstElement_ptr dat,
    uint64_t rowdim,
    uint64_t coldim,
    uint64_t nnz) [inline]
```



**11.1.3.195 sparse\_init()** [4/16]

```
template<class Field , class IndexT >
void sparse_init (
    const Field & F,
    Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
    const IndexT * row,
    const IndexT * col,
    typename Field::ConstElement_ptr dat,
    uint64_t rowdim,
    uint64_t coldim,
    uint64_t nnz) [inline]
```

**11.1.3.196 sparse\_delete()** [3/12]

```
template<class Field >
void sparse_delete (
    const Sparse< Field, SparseMatrix_t::CSR > & A) [inline]
```

**11.1.3.197 sparse\_delete()** [4/12]

```
template<class Field >
void sparse_delete (
    const Sparse< Field, SparseMatrix_t::CSR_ZO > & A) [inline]
```

**11.1.3.198 sparse\_print()** [1/3]

```
template<class Field >
std::ostream & sparse_print (
    std::ostream & os,
    const Sparse< Field, SparseMatrix_t::CSR > & A) [inline]
```

**11.1.3.199 sparse\_init()** [5/16]

```
template<class IndexT >
void sparse_init (
    const Givaro::Modular< Givaro::Integer > & F,
    Sparse< Givaro::Modular< Givaro::Integer >, SparseMatrix_t::CSR > & A,
    const IndexT * row,
    const IndexT * col,
    Givaro::Integer * dat,
    uint64_t rowdim,
    uint64_t coldim,
    uint64_t nnz) [inline]
```

**11.1.3.200 sparse\_init() [6/16]**

```
template<class IndexT >
void sparse_init (
    const Givaro::ZRing< Givaro::Integer > & F,
    Sparse< Givaro::ZRing< Givaro::Integer >, SparseMatrix_t::CSR_ZO > & A,
    const IndexT * row,
    const IndexT * col,
    Givaro::Integer * dat,
    uint64_t rowdim,
    uint64_t coldim,
    uint64_t nnz) [inline]
```

**11.1.3.201 sparse\_init() [7/16]**

```
template<class IndexT , size_t RECINT_SIZE>
void sparse_init (
    const Givaro::ZRing< RecInt::rmint< RECINT_SIZE > > & F,
    Sparse< Givaro::ZRing< RecInt::rmint< RECINT_SIZE > >, SparseMatrix_t::CSR_ZO >
& A,
    const IndexT * row,
    const IndexT * col,
    typename Givaro::ZRing< RecInt::rmint< RECINT_SIZE > >::Element_ptr dat,
    uint64_t rowdim,
    uint64_t coldim,
    uint64_t nnz) [inline]
```

**11.1.3.202 sparse\_init() [8/16]**

```
template<class IndexT , size_t RECINT_SIZE>
void sparse_init (
    const Givaro::ZRing< RecInt::rmint< RECINT_SIZE > > & F,
    Sparse< Givaro::ZRing< RecInt::rmint< RECINT_SIZE > >, SparseMatrix_t::CSR > &
A,
    const IndexT * row,
    const IndexT * col,
    typename Givaro::ZRing< RecInt::rmint< RECINT_SIZE > >::Element_ptr dat,
    uint64_t rowdim,
    uint64_t coldim,
    uint64_t nnz) [inline]
```

**11.1.3.203 sparse\_delete() [5/12]**

```
template<class Field >
void sparse_delete (
    const Sparse< Field, SparseMatrix_t::CSR_HYB > & A) [inline]
```

**11.1.3.204 sparse\_init()** [9/16]

```
template<class Field , class IndexT >
void sparse_init (
    const Field & F,
    Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
    const IndexT * row,
    const IndexT * col,
    typename Field::ConstElement_ptr dat,
    uint64_t rowdim,
    uint64_t coldim,
    uint64_t nnz) [inline]
```

**11.1.3.205 sparse\_init()** [10/16]

```
template<class Field , class IndexT >
void sparse_init (
    const Field & F,
    Sparse< Field, SparseMatrix_t::ELL > & A,
    const IndexT * row,
    const IndexT * col,
    typename Field::ConstElement_ptr dat,
    uint64_t rowdim,
    uint64_t coldim,
    uint64_t nnz) [inline]
```

**11.1.3.206 sparse\_init()** [11/16]

```
template<class Field , class IndexT >
void sparse_init (
    const Field & F,
    Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
    const IndexT * row,
    const IndexT * col,
    typename Field::ConstElement_ptr dat,
    uint64_t rowdim,
    uint64_t coldim,
    uint64_t nnz) [inline]
```

**11.1.3.207 sparse\_delete()** [6/12]

```
template<class Field >
void sparse_delete (
    const Sparse< Field, SparseMatrix_t::ELL > & A) [inline]
```

**11.1.3.208 sparse\_delete()** [7/12]

```
template<class Field >
void sparse_delete (
    const Sparse< Field, SparseMatrix_t::ELL_ZO > & A) [inline]
```

**11.1.3.209 sparse\_init()** [12/16]

```
template<class Field , class IndexT >
void sparse_init (
    const Field & F,
    Sparse< Field, SparseMatrix_t::ELL_simd > & A,
    const IndexT * row,
    const IndexT * col,
    typename Field::ConstElement_ptr dat,
    uint64_t rowdim,
    uint64_t coldim,
    uint64_t nnz) [inline]
```

**11.1.3.210 sparse\_init()** [13/16]

```
template<class Field , class IndexT >
void sparse_init (
    const Field & F,
    Sparse< Field, SparseMatrix_t::ELL_simd_ZO > & A,
    const IndexT * row,
    const IndexT * col,
    typename Field::ConstElement_ptr dat,
    uint64_t rowdim,
    uint64_t coldim,
    uint64_t nnz) [inline]
```

**11.1.3.211 sparse\_delete()** [8/12]

```
template<class Field >
void sparse_delete (
    const Sparse< Field, SparseMatrix_t::ELL_simd > & A) [inline]
```

**11.1.3.212 sparse\_delete()** [9/12]

```
template<class Field >
void sparse_delete (
    const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > & A) [inline]
```

**11.1.3.213 sparse\_print()** [2/3]

```
template<class Field >
void sparse_print (
    const Sparse< Field, SparseMatrix_t::ELL_simd > & A) [inline]
```

**11.1.3.214 sparse\_delete()** [10/12]

```
template<class Field >
void sparse_delete (
    const Sparse< Field, SparseMatrix_t::HYB_ZO > & A) [inline]
```

**11.1.3.215 sparse\_init()** [14/16]

```
template<class Field , class IndexT >
void sparse_init (
    const Field & F,
    Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
    const IndexT * row,
    const IndexT * col,
    typename Field::ConstElement_ptr dat,
    uint64_t rowdim,
    uint64_t coldim,
    uint64_t nnz) [inline]
```

**11.1.3.216 operator<<()**

```
template<typename _Field >
std::ostream & operator<< (
    std::ostream & os,
    const Sparse< _Field, SparseMatrix_t::HYB_ZO > & A)
```

**11.1.3.217 readSmsFormat()**

```
template<class Field , bool sorted = true, bool read_integer = false>
void readSmsFormat (
    const std::string & path,
    const Field & f,
    index_t *& row,
    index_t *& col,
    typename Field::Element_ptr & val,
    index_t & rowdim,
    index_t & coldim,
    uint64_t & nnz)
```

**11.1.3.218 readSprFormat()**

```
template<class Field >
void readSprFormat (
    const std::string & path,
    const Field & f,
    index_t *& row,
    index_t *& col,
    typename Field::Element_ptr & val,
    index_t & rowdim,
    index_t & coldim,
    uint64_t & nnz)
```

**11.1.3.219 getDataType()** [1/4]

```
template<class T >
std::enable_if< std::is_integral< T >::value, int > getDataType ()
```

**11.1.3.220** `getDataType()` [2/4]

```
template<class T >
std::enable_if< std::is_floating_point< T >::value, int > getDataType ()
```

**11.1.3.221** `getDataType()` [3/4]

```
template<class T >
std::enable_if< std::is_same< T, mpz_t >::value, int > getDataType ()
```

**11.1.3.222** `getDataType()` [4/4]

```
template<class T >
int getDataType ()
```

**11.1.3.223** `readMachineType()`

```
template<class Field >
void readMachineType (
    const Field & F,
    typename Field::Element & modulo,
    typename Field::Element_ptr val,
    std::ifstream & file,
    const uint64_t dims,
    const mask_t data_type,
    const mask_t field_desc)
```

**11.1.3.224** `readDnsFormat()`

```
template<class Field >
void readDnsFormat (
    const std::string & path,
    const Field & F,
    index_t & rowdim,
    index_t & coldim,
    typename Field::Element_ptr & val)
```

**11.1.3.225** `writeDnsFormat()`

```
template<class Field >
void writeDnsFormat (
    const std::string & path,
    const Field & F,
    const index_t & rowdim,
    const index_t & coldim,
    typename Field::Element_ptr A,
    index_t ldA)
```

**11.1.3.226 fspmv()** [2/2]

```
template<class Field >
void fspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::SELL_ZO > & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::ModularTag ) [inline]
```

**11.1.3.227 sparse\_delete()** [11/12]

```
template<class Field >
void sparse_delete (
    const Sparse< Field, SparseMatrix_t::SELL > & A) [inline]
```

**11.1.3.228 sparse\_delete()** [12/12]

```
template<class Field >
void sparse_delete (
    const Sparse< Field, SparseMatrix_t::SELL_ZO > & A) [inline]
```

**11.1.3.229 sparse\_print()** [3/3]

```
template<class Field >
void sparse_print (
    const Sparse< Field, SparseMatrix_t::SELL > & A) [inline]
```

**11.1.3.230 sparse\_init()** [15/16]

```
template<class Field , class IndexT >
void sparse_init (
    const Field & F,
    Sparse< Field, SparseMatrix_t::SELL > & A,
    const IndexT * row,
    const IndexT * col,
    typename Field::ConstElement_ptr dat,
    uint64_t rowdim,
    uint64_t coldim,
    uint64_t nnz,
    uint64_t sigma = 0) [inline]
```

**11.1.3.231 sparse\_init()** [16/16]

```
template<class Field , class IndexT >
void sparse_init (
    const Field & F,
    Sparse< Field, SparseMatrix_t::SELL_ZO > & A,
    const IndexT * row,
    const IndexT * col,
    typename Field::ConstElement_ptr dat,
    uint64_t rowdim,
    uint64_t coldim,
    uint64_t nnz) [inline]
```

**11.1.3.232 computeDeviation()**

```
template<class It >
double computeDeviation (
    It begin,
    It end)
```

**11.1.3.233 getStat()**

```
template<class Field >
StatsMatrix getStat (
    const Field & F,
    const index_t * row,
    const index_t * col,
    typename Field::ConstElement_ptr val,
    uint64_t rowdim,
    uint64_t coldim,
    uint64_t nnz)
```

**11.1.3.234 maxCardinality()**

```
template<class Field , class enable = void>
Field::Residu_t maxCardinality () [inline]
```

**11.1.3.235 maxCardinality< Givaro::Modular< int64\_t > >()**

```
template<>
uint64_t maxCardinality< Givaro::Modular< int64_t > > () [inline]
```

**11.1.3.236 maxCardinality< Givaro::Modular< int32\_t > >()**

```
template<>
uint32_t maxCardinality< Givaro::Modular< int32_t > > () [inline]
```

**11.1.3.237 minCardinality()**

```
template<class Field >
Field::Residu_t minCardinality () [inline]
```

**11.1.3.238 fflas\_delete() [1/4]**

```
template<>
void fflas_delete (
    FFPACK::rns_double_elt_ptr A) [inline]
```



**11.1.3.239 fflas\_delete()** [2/4]

```
template<>
void fflas_delete (
    FFPACK::rns_double_elt_cstptr A) [inline]
```

**11.1.3.240 fflas\_new()** [1/7]

```
template<>
FFPACK::rns_double_elt_ptr fflas_new (
    const FFPACK::RNSIntegerMod< FFPACK::rns_double > & F,
    const size_t m,
    const Alignment align) [inline]
```

**11.1.3.241 fflas\_new()** [2/7]

```
template<>
FFPACK::rns_double_elt_ptr fflas_new (
    const FFPACK::RNSIntegerMod< FFPACK::rns_double > & F,
    const size_t m,
    const size_t n,
    const Alignment align) [inline]
```

**11.1.3.242 finit\_rns()** [1/2]

```
template<typename RNS >
void finit_rns (
    const FFPACK::RNSIntegerMod< RNS > & F,
    const size_t m,
    const size_t n,
    size_t k,
    const Givaro::Integer * B,
    const size_t ldb,
    typename RNS::Element_ptr A)
```

**11.1.3.243 finit\_trans\_rns()**

```
template<typename RNS >
void finit_trans_rns (
    const FFPACK::RNSIntegerMod< RNS > & F,
    const size_t m,
    const size_t n,
    size_t k,
    const Givaro::Integer * B,
    const size_t ldb,
    typename RNS::Element_ptr A)
```

**11.1.3.244 fconvert\_rns()** [1/2]

```
template<typename RNS >
void fconvert_rns (
    const FFPACK::RNSIntegerMod< RNS > & F,
    const size_t m,
    const size_t n,
    Givaro::Integer alpha,
    Givaro::Integer * B,
    const size_t ldb,
    typename RNS::ConstElement_ptr A)
```

**11.1.3.245 fconvert\_trans\_rns()**

```
template<typename RNS >
void fconvert_trans_rns (
    const FFPACK::RNSIntegerMod< RNS > & F,
    const size_t m,
    const size_t n,
    Givaro::Integer alpha,
    Givaro::Integer * B,
    const size_t ldb,
    typename RNS::ConstElement_ptr A)
```

**11.1.3.246 fflas\_new()** [3/7]

```
template<>
FFPACK::rns_double_elt_ptr fflas_new (
    const FFPACK::RNSInteger< FFPACK::rns_double > & F,
    const size_t m,
    const Alignment align) [inline]
```

**11.1.3.247 fflas\_new()** [4/7]

```
template<>
FFPACK::rns_double_elt_ptr fflas_new (
    const FFPACK::RNSInteger< FFPACK::rns_double > & F,
    const size_t m,
    const size_t n,
    const Alignment align) [inline]
```

**11.1.3.248 finit\_rns()** [2/2]

```
template<typename RNS >
void finit_rns (
    const FFPACK::RNSInteger< RNS > & F,
    const size_t m,
    const size_t n,
    size_t k,
    const Givaro::Integer * B,
    const size_t ldb,
    typename FFPACK::RNSInteger< RNS >::Element_ptr A)
```

**11.1.3.249 fconvert\_rns()** [2/2]

```
template<typename RNS >
void fconvert_rns (
    const FFPACK::RNSInteger< RNS > & F,
    const size_t m,
    const size_t n,
    Givaro::Integer alpha,
    Givaro::Integer * B,
    const size_t ldb,
    typename FFPACK::RNSInteger< RNS >::ConstElement_ptr A)
```

**11.1.3.250 freduce()** [8/11]

```
template INST_OR_DECL void freduce (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t n,
    FFLAS_ELT * X,
    const size_t incX)
```

freduce  $x \leftarrow x \bmod F$ .

**Parameters**

$F$	field
$n$	size of the vectors
$X$	vector in $F$
$incX$	stride of $X$

**Bug** use cblas\_(d)scal when possible

**11.1.3.251 freduce()** [9/11]

```
template INST_OR_DECL void freduce (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t n,
    const FFLAS_ELT * Y,
    const size_t incY,
    FFLAS_ELT * X,
    const size_t incX)
```

freduce  $x \leftarrow y \bmod F$ .

**Parameters**

$F$	field
$n$	size of the vectors
$Y$	vector of Element
$incY$	stride of $Y$
$X$	vector in $F$
$incX$	stride of $X$

**Bug** use cblas\_(d)scal when possible

**11.1.3.252 finit()** [7/8]

```
template INST_OR_DECL void finit (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t n,
    const FFLAS_ELT * Y,
    const size_t incY,
    FFLAS_ELT * X,
    const size_t incX)
```

finit  $x \leftarrow y \bmod F$ .

**Parameters**

$F$	field
$n$	size of the vectors
$Y$	vector of OtherElement
$incY$	stride of Y
$X$	vector in F
$incX$	stride of X

**Bug** use cblas\_(d)scal when possible

**11.1.3.253 fconvert()** [3/3]

```
template INST_OR_DECL void fconvert (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t n,
    FFLAS_ELT * X,
    const size_t incX,
    const FFLAS_ELT * Y,
    const size_t incY)
```

fconvert  $x \leftarrow y \bmod F$ .

**Parameters**

$F$	field
$n$	size of the vectors
$Y$	vector of F
$incY$	stride of Y
$X$	vector in OtherElement
$incX$	stride of X

**Bug** use cblas\_(d)scal when possible

**11.1.3.254 fnegin()** [3/4]

```
template INST_OR_DECL void fnegin (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t n,
    FFLAS_ELT * X,
    const size_t incX)
```

fnegin  $x \leftarrow -x$ .

## Parameters

$F$	field
$n$	size of the vectors
$X$	vector in $F$
$incX$	stride of $X$

**Bug** use `cblas_(d)scal` when possible

11.1.3.255 `fneg()` [3/4]

```
template INST_OR_DECL void fneg (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t n,
    const FFLAS_ELT * Y,
    const size_t incY,
    FFLAS_ELT * X,
    const size_t incX)
```

`fneg`  $x \leftarrow -y$ .

## Parameters

$F$	field
$n$	size of the vectors
$X$	vector in $F$
$incX$	stride of $X$
$Y$	vector in $F$
$incY$	stride of $Y$

**Bug** use `cblas_(d)scal` when possible

11.1.3.256 `fzero()` [4/5]

```
template INST_OR_DECL void fzero (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t n,
    FFLAS_ELT * X,
    const size_t incX)
```

`fzero` :  $A \leftarrow 0$ .

## Parameters

$F$	field
$n$	number of elements to zero
$X$	vector in $F$
$incX$	stride of $X$

**11.1.3.257 fiszero()** [3/4]

```
template INST_OR_DECL bool fiszero (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t n,
    const FFLAS_ELT * X,
    const size_t incX)
```

**fiszero** : test  $X = 0$ .

**Parameters**

$F$	field
$n$	vector dimension
$X$	vector in $F$
$incX$	increment of $X$

**11.1.3.258 fequal()** [3/4]

```
template INST_OR_DECL bool fequal (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t n,
    const FFLAS_ELT * X,
    const size_t incX,
    const FFLAS_ELT * Y,
    const size_t incY)
```

**fequal** : test  $X = Y$ .

**Parameters**

$F$	field
$n$	vector dimension
$X$	vector in $F$
$incX$	increment of $X$
$Y$	vector in $F$
$incY$	increment of $Y$

**11.1.3.259 fassign()** [9/10]

```
template INST_OR_DECL void fassign (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t N,
    const FFLAS_ELT * Y,
    const size_t incY,
    FFLAS_ELT * X,
    const size_t incX)
```

**fassign** :  $x \leftarrow y$ .

$X$  is preallocated

**Todo** variant for triagular matrix

## Parameters

	$F$	field
	$N$	size of the vectors
out	$X$	vector in $F$
	$incX$	stride of $X$
in	$Y$	vector in $F$
	$incY$	stride of $Y$

## 11.1.3.260 fscaln() [9/10]

```
template INST_OR_DECL void fscaln (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t n,
    const FFLAS_ELT alpha,
    FFLAS_ELT * X,
    const size_t incX)
```

$\text{fscaln } x \leftarrow \alpha \cdot x.$

## Parameters

$F$	field
$n$	size of the vectors
$alpha$	scalar
$X$	vector in $F$
$incX$	stride of $X$

**Bug** use `cblas_(d)scal` when possible

**Todo** check if comparison with  $\pm 1, 0$  is necessary.

## 11.1.3.261 fscal() [9/10]

```
template INST_OR_DECL void fscal (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t n,
    const FFLAS_ELT alpha,
    const FFLAS_ELT * X,
    const size_t incX,
    FFLAS_ELT * Y,
    const size_t incY)
```

$\text{fscal } y \leftarrow \alpha \cdot x.$

## Parameters

	$F$	field
	$n$	size of the vectors

	<i>alpha</i>	scalar
in	<i>X</i>	vector in $F$
	<i>incX</i>	stride of <i>X</i>
out	<i>Y</i>	vector in $F$
	<i>incY</i>	stride of <i>Y</i>

**Bug** use `cblas_(d)scal` when possible

**Todo** check if comparison with  $\pm 1, 0$  is necessary.

### 11.1.3.262 faxpy() [5/6]

```
template INST_OR_DECL void faxpy (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t N,
    const FFLAS_ELT alpha,
    const FFLAS_ELT * X,
    const size_t incX,
    FFLAS_ELT * Y,
    const size_t incY)
```

$\text{faxpy} : y \leftarrow \alpha \cdot x + y.$

#### Parameters

	<i>F</i>	field
	<i>N</i>	size of the vectors
	<i>alpha</i>	scalar
in	<i>X</i>	vector in $F$
	<i>incX</i>	stride of <i>X</i>
in, out	<i>Y</i>	vector in $F$
	<i>incY</i>	stride of <i>Y</i>

### 11.1.3.263 fdot() [10/11]

```
template INST_OR_DECL FFLAS_ELT fdot (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t N,
    const FFLAS_ELT * X,
    const size_t incX,
    const FFLAS_ELT * Y,
    const size_t incY)
```

$\text{faxpby} : y \leftarrow \alpha \cdot x + \beta \cdot y.$



## Parameters

	$F$	field
	$N$	size of the vectors
	$\alpha$	scalar
in	$X$	vector in $F$
	$incX$	stride of $X$
	$\beta$	scalar
in, out	$Y$	vector in $F$
	$incY$	stride of $Y$

## Note

this is a catlas function

fdot: dot product  $x^T y$ .

## Parameters

$F$	field
$N$	size of the vectors
$X$	vector in $F$
$incX$	stride of $X$
$Y$	vector in $F$
$incY$	stride of $Y$

## 11.1.3.264 fswap() [2/2]

```
template INST_OR_DECL void fswap (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t N,
    FFLAS_ELT * X,
    const size_t incX,
    FFLAS_ELT * Y,
    const size_t incY)
```

fswap:  $X \leftrightarrow Y$ .

**Bug** use cblas\_dswap when double

## Parameters

$F$	field
$N$	size of the vectors
$X$	vector in $F$
$incX$	stride of $X$
$Y$	vector in $F$
$incY$	stride of $Y$

**11.1.3.265 fadd()** [5/8]

```
template INST_OR_DECL void fadd (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t N,
    const FFLAS_ELT * A,
    const size_t inca,
    const FFLAS_ELT * B,
    const size_t incb,
    FFLAS_ELT * C,
    const size_t incc)
```

**11.1.3.266 fsub()** [3/4]

```
template INST_OR_DECL void fsub (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t N,
    const FFLAS_ELT * A,
    const size_t inca,
    const FFLAS_ELT * B,
    const size_t incb,
    FFLAS_ELT * C,
    const size_t incc)
```

**11.1.3.267 faddin()** [4/5]

```
template INST_OR_DECL void faddin (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t N,
    const FFLAS_ELT * B,
    const size_t incb,
    FFLAS_ELT * C,
    const size_t incc)
```

**11.1.3.268 fadd()** [6/8]

```
template INST_OR_DECL void fadd (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t N,
    const FFLAS_ELT * A,
    const size_t inca,
    const FFLAS_ELT alpha,
    const FFLAS_ELT * B,
    const size_t incb,
    FFLAS_ELT * C,
    const size_t incc)
```

**11.1.3.269 fassign()** [10/10]

```
template INST_OR_DECL void fassign (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t m,
    const size_t n,
    const FFLAS_ELT * B,
    const size_t ldb,
    FFLAS_ELT * A,
    const size_t lda)
```

fassign :  $A \leftarrow B$ .

**Parameters**

$F$	field
$m$	number of rows to copy
$n$	number of cols to copy
$A$	matrix in $F$
$lda$	stride of $A$
$B$	vector in $F$
$ldb$	stride of $B$

**11.1.3.270 fzero()** [5/5]

```
template INST_OR_DECL void fzero (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t m,
    const size_t n,
    FFLAS_ELT * A,
    const size_t lda)
```

fzero :  $A \leftarrow 0$ .

**Parameters**

$F$	field
$m$	number of rows to zero
$n$	number of cols to zero
$A$	matrix in $F$
$lda$	stride of $A$

**Warning**

may be buggy if Element is larger than int

**11.1.3.271 fequal()** [4/4]

```
template INST_OR_DECL bool fequal (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t m,
    const size_t n,
    const FFLAS_ELT * A,
    const size_t lda,
    const FFLAS_ELT * B,
    const size_t ldb)
```

fequal : test  $A = B$ .

**Parameters**

$F$	field
$m$	row dimension
$n$	column dimension
$A$	m x n matrix in $F$
$lda$	leading dimension of A
$B$	m x n matrix in $F$
$ldb$	leading dimension of B

**11.1.3.272 fiszero()** [4/4]

```
template INST_OR_DECL bool fiszero (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t m,
    const size_t n,
    const FFLAS_ELT * A,
    const size_t lda)
```

fiszero : test  $A = 0$ .

**Parameters**

$F$	field
$m$	row dimension
$n$	column dimension
$A$	m x n matrix in $F$
$lda$	leading dimension of A

**11.1.3.273 fidentity()** [3/4]

```
template INST_OR_DECL void fidentity (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t m,
    const size_t n,
    FFLAS_ELT * A,
    const size_t lda,
    const FFLAS_ELT & d)
```

creates a diagonal matrix

**11.1.3.274 fidentity()** [4/4]

```
template INST_OR_DECL void fidentity (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t m,
    const size_t n,
    FFLAS_ELT * A,
    const size_t lda)
```

creates a diagonal matrix

**11.1.3.275 freduce()** [10/11]

```
template INST_OR_DECL void freduce (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t m,
    const size_t n,
    FFLAS_ELT * A,
    const size_t lda)
```

freduce  $A \leftarrow A \bmod F$ .

**Parameters**

$F$	field
$m$	number of rows
$n$	number of cols
$A$	matrix in F
$lda$	stride of A

**11.1.3.276 freduce()** [11/11]

```
template INST_OR_DECL void freduce (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t m,
    const size_t n,
    const FFLAS_ELT * B,
    const size_t ldb,
    FFLAS_ELT * A,
    const size_t lda)
```

freduce  $A \leftarrow B \bmod F$ .

**Parameters**

$F$	field
$m$	number of rows
$n$	number of cols
$A$	matrix in F
$lda$	stride of A
$B$	matrix in Element
$ldb$	stride of B

**11.1.3.277 finit()** [8/8]

```
template INST_OR_DECL void finit (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t m,
    const size_t n,
    const FFLAS_ELT * B,
    const size_t ldb,
    FFLAS_ELT * A,
    const size_t lda)
```

$\text{finit } A \leftarrow B \bmod F.$

**Parameters**

$F$	field
$m$	number of rows
$n$	number of cols
$A$	matrix in $F$
$lda$	stride of $A$
$B$	matrix in $F$
$ldb$	stride of $B$

**11.1.3.278 fnegin()** [4/4]

```
template INST_OR_DECL void fnegin (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t m,
    const size_t n,
    FFLAS_ELT * A,
    const size_t lda)
```

$\text{fnegin } A \leftarrow -A.$

**Parameters**

$F$	field
$m$	number of rows
$n$	number of cols
$A$	matrix in $F$
$lda$	stride of $A$

**11.1.3.279 fneg()** [4/4]

```
template INST_OR_DECL void fneg (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t m,
    const size_t n,
    const FFLAS_ELT * B,
    const size_t ldb,
    FFLAS_ELT * A,
    const size_t lda)
```

$\text{fneg } A \leftarrow -B.$

## Parameters

$F$	field
$m$	number of rows
$n$	number of cols
$A$	matrix in $F$
$lda$	stride of $A$

11.1.3.280 `fscaln()` [10/10]

```
template INST_OR_DECL void fscaln (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t m,
    const size_t n,
    const FFLAS_ELT alpha,
    FFLAS_ELT * A,
    const size_t lda)
```

$\text{fscaln } A \leftarrow a \cdot A.$

## Parameters

$F$	field
$m$	number of rows
$n$	number of cols
$alpha$	homotecie scalar
$A$	matrix in $F$
$lda$	stride of $A$

11.1.3.281 `fscale()` [10/10]

```
template INST_OR_DECL void fscale (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t m,
    const size_t n,
    const FFLAS_ELT alpha,
    const FFLAS_ELT * A,
    const size_t lda,
    FFLAS_ELT * B,
    const size_t ldb)
```

$\text{fscale } B \leftarrow a \cdot A.$

## Parameters

	$F$	field
	$m$	number of rows
	$n$	number of cols
	$alpha$	homotecie scalar
in	$A$	matrix in $F$
	$lda$	stride of $A$
out	$B$	matrix in $F$
	$ldb$	stride of $B$

**11.1.3.282 faxpy()** [6/6]

```
template INST_OR_DECL void faxpy (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t m,
    const size_t n,
    const FFLAS_ELT alpha,
    const FFLAS_ELT * X,
    const size_t ldx,
    FFLAS_ELT * Y,
    const size_t ldy)
```

$\text{faxpy} : y \leftarrow \alpha \cdot x + y.$

**Parameters**

	$F$	field
	$m$	row dimension
	$n$	column dimension
	$\alpha$	scalar
in	$X$	vector in F
	$ldx$	leading dimension of X
in, out	$Y$	vector in F
	$ldy$	leading dimension of Y

**11.1.3.283 fmove()** [2/2]

```
template INST_OR_DECL void fmove (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t m,
    const size_t n,
    FFLAS_ELT * A,
    const size_t lda,
    FFLAS_ELT * B,
    const size_t ldb)
```

$\text{faxpby} : y \leftarrow \alpha \cdot x + \beta \cdot y.$

**Parameters**

	$F$	field
	$m$	row dimension
	$n$	column dimension
	$\alpha$	scalar
in	$X$	vector in F
	$ldx$	leading dimension of X
	$\beta$	scalar
in, out	$Y$	vector in F
	$ldy$	leading dimension of Y

**Note**

this is a catlas function

$\text{fmove} : A \leftarrow B \text{ and } B \leftarrow 0.$



## Parameters

<i>F</i>	field
<i>m</i>	number of rows to copy
<i>n</i>	number of cols to copy
<i>A</i>	matrix in <i>F</i>
<i>lda</i>	stride of <i>A</i>
<i>B</i>	vector in <i>F</i>
<i>ldb</i>	stride of <i>B</i>

**11.1.3.284 fadd()** [7/8]

```
template INST_OR_DECL void fadd (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t M,
    const size_t N,
    const FFLAS_ELT * A,
    const size_t lda,
    const FFLAS_ELT * B,
    const size_t ldb,
    FFLAS_ELT * C,
    const size_t ldc)
```

fadd : matrix addition.

Computes  $C = A + B$ .

## Parameters

<i>F</i>	field
<i>M</i>	rows
<i>N</i>	cols
<i>A</i>	dense matrix of size $M \times N$
<i>lda</i>	leading dimension of <i>A</i>
<i>B</i>	dense matrix of size $M \times N$
<i>ldb</i>	leading dimension of <i>B</i>
<i>C</i>	dense matrix of size $M \times N$
<i>ldc</i>	leading dimension of <i>C</i>

**11.1.3.285 fsub()** [4/4]

```
template INST_OR_DECL void fsub (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t M,
    const size_t N,
    const FFLAS_ELT * A,
    const size_t lda,
    const FFLAS_ELT * B,
    const size_t ldb,
    FFLAS_ELT * C,
    const size_t ldc)
```

fsub : matrix subtraction.

Computes  $C = A - B$ .

## Parameters

<i>F</i>	field
<i>M</i>	rows
<i>N</i>	cols
<i>A</i>	dense matrix of size $M \times N$
<i>lda</i>	leading dimension of A
<i>B</i>	dense matrix of size $M \times N$
<i>ldb</i>	leading dimension of B
<i>C</i>	dense matrix of size $M \times N$
<i>ldc</i>	leading dimension of C

**11.1.3.286 fsubin()** [3/3]

```
template INST_OR_DECL void fsubin (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t M,
    const size_t N,
    const FFLAS_ELT * B,
    const size_t ldb,
    FFLAS_ELT * C,
    const size_t ldc)
```

fsubin C = C - B

**11.1.3.287 fadd()** [8/8]

```
template INST_OR_DECL void fadd (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t M,
    const size_t N,
    const FFLAS_ELT * A,
    const size_t lda,
    const FFLAS_ELT alpha,
    const FFLAS_ELT * B,
    const size_t ldb,
    FFLAS_ELT * C,
    const size_t ldc)
```

fadd : matrix addition with scaling.

Computes  $C = A + \alpha B$ .

## Parameters

<i>F</i>	field
<i>M</i>	rows
<i>N</i>	cols
<i>A</i>	dense matrix of size $M \times N$
<i>lda</i>	leading dimension of A
<i>alpha</i>	some scalar
<i>B</i>	dense matrix of size $M \times N$
<i>ldb</i>	leading dimension of B
<i>C</i>	dense matrix of size $M \times N$
<i>ldc</i>	leading dimension of C

**11.1.3.288 faddin()** [5/5]

```
template INST_OR_DECL void faddin (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t M,
    const size_t N,
    const FFLAS_ELT * B,
    const size_t ldb,
    FFLAS_ELT * C,
    const size_t ldc)
```

faddin

**11.1.3.289 fgemv()** [17/19]

```
template INST_OR_DECL FFLAS_ELT * fgemv (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS_TRANSPOSE TransA,
    const size_t M,
    const size_t N,
    const FFLAS_ELT alpha,
    const FFLAS_ELT * A,
    const size_t lda,
    const FFLAS_ELT * X,
    const size_t incX,
    const FFLAS_ELT beta,
    FFLAS_ELT * Y,
    const size_t incY)
```

finite prime FFLAS\_FIELD<FFLAS\_ELT> GEneral Matrix Vector multiplication.

Computes  $Y \leftarrow \alpha \text{op}(A)X + \beta Y$ .

**Parameters**

	$F$	field
	$TransA$	if $TransA == FflasTrans$ then $\text{op}(A) = A^t$ .
	$M$	rows
	$N$	cols
	$alpha$	scalar
	$A$	dense matrix of size $M \times N$
	$lda$	leading dimension of $A$
	$X$	dense vector of size $N$
	$incX$	stride of $X$
	$beta$	scalar
out	$Y$	dense vector of size $M$
	$incY$	stride of $Y$

**11.1.3.290 fger()** [12/12]

```
template INST_OR_DECL void fger (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t M,
    const size_t N,
    const FFLAS_ELT alpha,
    const FFLAS_ELT * x,
    const size_t incx,
    const FFLAS_ELT * y,
    const size_t incy,
    FFLAS_ELT * A,
    const size_t lda)
```

fger: rank one update of a general matrix

Computes  $A \leftarrow \alpha x y^T + A$

**Parameters**

	$F$	field
	$M$	rows
	$N$	cols
	$\alpha$	scalar
in, out	$A$	dense matrix of size $M \times N$ and leading dimension $lda$
	$lda$	leading dimension of $A$
	$x$	dense vector of size $M$
	$incx$	stride of $X$
	$y$	dense vector of size $N$
	$incy$	stride of $Y$

**11.1.3.291 ftrsv()** [2/2]

```
template INST_OR_DECL void ftrsv (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS_UPLO Uplo,
    const FFLAS_TRANSPOSE TransA,
    const FFLAS_DIAG Diag,
    const size_t N,
    const FFLAS_ELT * A,
    const size_t lda,
    FFLAS_ELT * X,
    int incX)
```

ftrsv: TRIangular System solve with Vector Computes  $X \leftarrow \text{op}(A^{-1})X$

**Parameters**

$F$	field
$X$	vector of size $N$ on a field $F$
$incX$	stride of $X$
$A$	a matrix of leading dimension $lda$ and size $N$
$lda$	leading dimension of $A$

## Parameters

<i>N</i>	number of rows or columns of A according to TransA
<i>TransA</i>	if TransA==FflasTrans then $\text{op}(A) = A^t$ .
<i>Diag</i>	if Diag==FflasUnit then A is unit.
<i>Uplo</i>	if Uplo==FflasUpper then A is upper triangular

## 11.1.3.292 ftrsm() [9/9]

```
template INST_OR_DECL void ftrsm (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS_SIDE Side,
    const FFLAS_UPLO Uplo,
    const FFLAS_TRANSPOSE TransA,
    const FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    const FFLAS_ELT alpha,
    const FFLAS_ELT * A,
    const size_t lda,
    FFLAS_ELT * B,
    const size_t ldb)
```

ftrsm: **T**Riangular **S**ystem solve with **M**atrix.

Computes  $B \leftarrow \alpha \text{op}(A^{-1})B$  or  $B \leftarrow \alpha B \text{op}(A^{-1})$ .

## Parameters

<i>F</i>	field
<i>Side</i>	if Side==FflasLeft then $B \leftarrow \alpha \text{op}(A^{-1})B$ is computed.
<i>Uplo</i>	if Uplo==FflasUpper then A is upper triangular
<i>TransA</i>	if TransA==FflasTrans then $\text{op}(A) = A^t$ .
<i>Diag</i>	if Diag==FflasUnit then A is unit.
<i>M</i>	rows of B
<i>N</i>	cols of B
<i>alpha</i>	scalar
<i>A</i>	triangular invertible matrix. If Side==FflasLeft then A is $N \times N$ , otherwise A is $M \times M$
<i>lda</i>	leading dim of A
<i>B</i>	matrix of size MxN
<i>ldb</i>	leading dim of B

**Bug**  $\alpha$  must be non zero.

**11.1.3.293 ftrmm()** [3/3]

```
template INST_OR_DECL void ftrmm (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS_SIDE Side,
    const FFLAS_UPLO Uplo,
    const FFLAS_TRANSPOSE TransA,
    const FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    const FFLAS_ELT alpha,
    const FFLAS_ELT * A,
    const size_t lda,
    FFLAS_ELT * B,
    const size_t ldb)
```

ftrmm: **TR**iangular **M**atrix **M**ultiply.

Computes  $B \leftarrow \alpha \text{op}(A)B$  or  $B \leftarrow \alpha B \text{op}(A)$ .

**Parameters**

<i>F</i>	field
<i>Side</i>	if Side==FflasLeft then $B \leftarrow \alpha \text{op}(A)B$ is computed.
<i>Uplo</i>	if Uplo==FflasUpper then A is upper triangular
<i>TransA</i>	if TransA==FflasTrans then $\text{op}(A) = A^t$ .
<i>Diag</i>	if Diag==FflasUnit then A is implicitly unit.
<i>M</i>	rows of B
<i>N</i>	cols of B
<i>alpha</i>	scalar
<i>A</i>	triangular matrix. If Side==FflasLeft then A is $N \times N$ , otherwise A is $M \times M$
<i>lda</i>	leading dim of A
<i>B</i>	matrix of size MxN
<i>ldb</i>	leading dim of B

**11.1.3.294 fgemm()** [20/23]

```
template INST_OR_DECL FFLAS_ELT * fgemm (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const FFLAS_ELT alpha,
    const FFLAS_ELT * A,
    const size_t lda,
    const FFLAS_ELT * B,
    const size_t ldb,
    const FFLAS_ELT beta,
    FFLAS_ELT * C,
    const size_t ldc)
```

fgemm: **F**ield **G**eneral **M**atrix **M**ultiply.

Computes  $C = \alpha \text{op}(A) \times \text{op}(B) + \beta C$  Automatically set Winograd recursion level

## Parameters

$F$	field.
$ta$	if $ta == \text{FflasTrans}$ then $\text{op}(A) = A^t$ , else $\text{op}(A) = A$ ,
$tb$	same for matrix B
$m$	see A
$n$	see B
$k$	see A
$\alpha$	scalar
$\beta$	scalar
$A$	$\text{op}(A)$ is $m \times k$
$B$	$\text{op}(B)$ is $k \times n$
$C$	$C$ is $m \times n$
$lda$	leading dimension of A
$ldb$	leading dimension of B
$ldc$	leading dimension of C
$w$	recursive levels of Winograd's algorithm are used. No argument (or -1) does auto computation of $w$ .

## Warning

$\alpha$  must be invertible

## 11.1.3.295 fgemm() [21/23]

```
template INST_OR_DECL FFLAS_ELT * fgemm (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const FFLAS_ELT alpha,
    const FFLAS_ELT * A,
    const size_t lda,
    const FFLAS_ELT * B,
    const size_t ldb,
    const FFLAS_ELT beta,
    FFLAS_ELT * C,
    const size_t ldc,
    const ParSeqHelper::Sequential seq)
```

## 11.1.3.296 fgemm() [22/23]

```
template INST_OR_DECL FFLAS_ELT * fgemm (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
```

```

    const FFLAS_ELT alpha,
    const FFLAS_ELT * A,
    const size_t lda,
    const FFLAS_ELT * B,
    const size_t ldb,
    const FFLAS_ELT beta,
    FFLAS_ELT * C,
    const size_t ldc,
    const ParSeqHelper::Parallel< CuttingStrategy::Recursive, StrategyParameter::TwoDAdaptive
> par)

```

### 11.1.3.297 fgemm() [23/23]

```

template INST_OR_DECL FFLAS_ELT * fgemm (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const FFLAS_ELT alpha,
    const FFLAS_ELT * A,
    const size_t lda,
    const FFLAS_ELT * B,
    const size_t ldb,
    const FFLAS_ELT beta,
    FFLAS_ELT * C,
    const size_t ldc,
    const ParSeqHelper::Parallel< CuttingStrategy::Block, StrategyParameter::Threads
> par)

```

### 11.1.3.298 fsquare() [6/6]

```

template INST_OR_DECL FFLAS_ELT * fsquare (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS_TRANSPOSE ta,
    const size_t n,
    const FFLAS_ELT alpha,
    const FFLAS_ELT * A,
    const size_t lda,
    const FFLAS_ELT beta,
    FFLAS_ELT * C,
    const size_t ldc)

```

fsquare: Squares a matrix.

compute  $C \leftarrow \alpha \text{op}(A) \text{op}(A) + \beta C$  over a `FFLAS_FIELD <FFLAS_ELT> F` Avoid the conversion of B

#### Parameters

<i>ta</i>	if <code>ta==FflasTrans</code> , $\text{op}(A) = A^T$ .
<i>F</i>	field
<i>n</i>	size of A
<i>alpha</i>	scalar



## Parameters

<i>beta</i>	scalar
<i>A</i>	dense matrix of size $n \times n$
<i>lda</i>	leading dimension of <i>A</i>
<i>C</i>	dense matrix of size $n \times n$
<i>ldc</i>	leading dimension of <i>C</i>

**11.1.3.299 BlockCuts()** [1/2]

```
template<class Cut = CuttingStrategy::Block, class Strat = StrategyParameter::Threads>
void BlockCuts (
    size_t & RBLOCKSIZE,
    size_t & CBLOCKSIZE,
    const size_t m,
    const size_t n,
    const size_t numthreads) [inline]
```

**11.1.3.300 BlockCuts< CuttingStrategy::Single, StrategyParameter::Threads >()**

```
template<>
void BlockCuts< CuttingStrategy::Single, StrategyParameter::Threads > (
    size_t & RBLOCKSIZE,
    size_t & CBLOCKSIZE,
    const size_t m,
    const size_t n,
    const size_t numthreads) [inline]
```

**11.1.3.301 BlockCuts< CuttingStrategy::Row, StrategyParameter::Fixed >()**

```
template<>
void BlockCuts< CuttingStrategy::Row, StrategyParameter::Fixed > (
    size_t & RBLOCKSIZE,
    size_t & CBLOCKSIZE,
    const size_t m,
    const size_t n,
    const size_t numthreads) [inline]
```

**11.1.3.302 BlockCuts< CuttingStrategy::Row, StrategyParameter::Grain >()**

```
template<>
void BlockCuts< CuttingStrategy::Row, StrategyParameter::Grain > (
    size_t & RBLOCKSIZE,
    size_t & CBLOCKSIZE,
    const size_t m,
    const size_t n,
    const size_t grainsize) [inline]
```

**11.1.3.303 BlockCuts< CuttingStrategy::Block, StrategyParameter::Grain >()**

```
template<>
void BlockCuts< CuttingStrategy::Block, StrategyParameter::Grain > (
    size_t & RBLOCKSIZE,
    size_t & CBLOCKSIZE,
    const size_t m,
    const size_t n,
    const size_t grainsize) [inline]
```

**11.1.3.304 BlockCuts< CuttingStrategy::Column, StrategyParameter::Fixed >()**

```
template<>
void BlockCuts< CuttingStrategy::Column, StrategyParameter::Fixed > (
    size_t & RBLOCKSIZE,
    size_t & CBLOCKSIZE,
    const size_t m,
    const size_t n,
    const size_t numthreads) [inline]
```

**11.1.3.305 BlockCuts< CuttingStrategy::Column, StrategyParameter::Grain >()**

```
template<>
void BlockCuts< CuttingStrategy::Column, StrategyParameter::Grain > (
    size_t & RBLOCKSIZE,
    size_t & CBLOCKSIZE,
    const size_t m,
    const size_t n,
    const size_t grainsize) [inline]
```

**11.1.3.306 BlockCuts< CuttingStrategy::Block, StrategyParameter::Fixed >()**

```
template<>
void BlockCuts< CuttingStrategy::Block, StrategyParameter::Fixed > (
    size_t & RBLOCKSIZE,
    size_t & CBLOCKSIZE,
    const size_t m,
    const size_t n,
    const size_t numthreads) [inline]
```

**11.1.3.307 BlockCuts< CuttingStrategy::Row, StrategyParameter::Threads >()**

```
template<>
void BlockCuts< CuttingStrategy::Row, StrategyParameter::Threads > (
    size_t & RBLOCKSIZE,
    size_t & CBLOCKSIZE,
    const size_t m,
    const size_t n,
    const size_t numthreads) [inline]
```

**11.1.3.308 BlockCuts< CuttingStrategy::Column, StrategyParameter::Threads >()**

```
template<>
void BlockCuts< CuttingStrategy::Column, StrategyParameter::Threads > (
    size_t & RBLOCKSIZE,
    size_t & CBLOCKSIZE,
    const size_t m,
    const size_t n,
    const size_t numthreads) [inline]
```

**11.1.3.309 BlockCuts< CuttingStrategy::Block, StrategyParameter::Threads >()**

```
template<>
void BlockCuts< CuttingStrategy::Block, StrategyParameter::Threads > (
    size_t & RBLOCKSIZE,
    size_t & CBLOCKSIZE,
    const size_t m,
    const size_t n,
    const size_t numthreads) [inline]
```

**11.1.3.310 BlockCuts() [2/2]**

```
template<class Cut = CuttingStrategy::Block, class Param = StrategyParameter::Threads>
void BlockCuts (
    size_t & rowBlockSize,
    size_t & colBlockSize,
    size_t & lastRBS,
    size_t & lastCBS,
    size_t & changeRBS,
    size_t & changeCBS,
    size_t & numRowsBlock,
    size_t & numColBlock,
    size_t m,
    size_t n,
    const size_t numthreads) [inline]
```

**11.1.3.311 pfzero()**

```
template<class Field >
void pfzero (
    const Field & F,
    size_t m,
    size_t n,
    typename Field::Element_ptr C,
    size_t BS = 0)
```

**11.1.3.312 pfrand()**

```
template<class Field , class RandIter >
void pfrand (
    const Field & F,
    RandIter & G,
    size_t m,
    size_t n,
    typename Field::Element_ptr C,
    size_t BS = 0)
```

**11.1.3.313 fdot()** [11/11]

```
template<class Field , class Cut , class Param >
Field::Element & fdot (
    const Field & F,
    const size_t N,
    typename Field::ConstElement_ptr x,
    const size_t incx,
    typename Field::ConstElement_ptr y,
    const size_t incy,
    typename Field::Element & d,
    const ParSeqHelper::Parallel< Cut, Param > par) [inline]
```

**11.1.3.314 pfgemm()** [2/7]

```
template<class Field , class AlgoT , class FieldTrait >
Field::Element * pfgemm (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    const typename Field::ConstElement_ptr A,
    const size_t lda,
    const typename Field::ConstElement_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element * C,
    const size_t ldc,
    MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Block,
StrategyParameter::Threads > > & H)
```

**11.1.3.315 pfgemm()** [3/7]

```
template<class Field , class AlgoT , class FieldTrait >
Field::Element * pfgemm (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    const typename Field::ConstElement_ptr AA,
    const size_t lda,
    const typename Field::ConstElement_ptr BB,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element * C,
    const size_t ldc,
    MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive,
StrategyParameter::ThreeDAdaptive > > & H)
```

**11.1.3.316 pfgemm() [4/7]**

```
template<class Field , class AlgoT , class FieldTrait >
Field::Element * pfgemm (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    const typename Field::ConstElement_ptr AA,
    const size_t lda,
    const typename Field::ConstElement_ptr BB,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element * C,
    const size_t ldc,
    MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive,
StrategyParameter::TwoDAdaptive > > & H)
```

**11.1.3.317 pfgemm() [5/7]**

```
template<class Field , class AlgoT , class FieldTrait >
Field::Element * pfgemm (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    const typename Field::ConstElement_ptr AA,
    const size_t lda,
    const typename Field::ConstElement_ptr BB,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element * C,
    const size_t ldc,
    MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive,
StrategyParameter::TwoD > > & H)
```

**11.1.3.318 pfgemm() [6/7]**

```
template<class Field , class AlgoT , class FieldTrait >
Field::Element_ptr pfgemm (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    const typename Field::ConstElement_ptr A,
```

```

    const size_t lda,
    const typename Field::ConstElement_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive,
StrategyParameter::ThreeD > > & H)

```

#### 11.1.3.319 pfgemm() [7/7]

```

template<class Field , class AlgoT , class FieldTrait >
Field::Element * pfgemm (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    const typename Field::ConstElement_ptr A,
    const size_t lda,
    const typename Field::ConstElement_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive,
StrategyParameter::ThreeDInPlace > > & H)

```

#### 11.1.3.320 fgemv() [18/19]

```

template<class Field , class AlgoT , class FieldTrait >
Field::Element_ptr fgemv (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const size_t m,
    const size_t n,
    const typename Field::Element alpha,
    const typename Field::ConstElement_ptr A,
    const size_t lda,
    const typename Field::ConstElement_ptr X,
    const size_t incX,
    const typename Field::Element beta,
    typename Field::Element_ptr Y,
    const size_t incY,
    MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive,
StrategyParameter::Threads > > & H)

```

#### 11.1.3.321 fgemv() [19/19]

```

template<class Field , class AlgoT , class FieldTrait , class Cut >
Field::Element_ptr fgemv (

```

```

    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const size_t m,
    const size_t n,
    const typename Field::Element alpha,
    const typename Field::ConstElement_ptr A,
    const size_t lda,
    const typename Field::ConstElement_ptr X,
    const size_t incX,
    const typename Field::Element beta,
    typename Field::Element_ptr Y,
    const size_t incY,
    MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Row,
Cut > > & H)

```

### 11.1.3.322 parseArguments()

```

void parseArguments (
    int argc,
    char ** argv,
    Argument * args,
    bool printDefaults = true)

```

### 11.1.3.323 getArgumentValue()

```

char * getArgumentValue (
    int argc,
    char ** argv,
    int i)

```

Get the value of an argument and avoid core dump when no value was given after an argument.

#### Parameters

<i>argv</i>	argument value list
<i>i</i>	argument index

#### Returns

char\* argument value

### 11.1.3.324 writeCommandString()

```

std::ostream & writeCommandString (
    std::ostream & os,
    Argument * args,
    const char * programName = nullptr)

```

writes the values of all arguments, preceded by the programName

**11.1.3.325 WriteMatrix()** [1/2]

```
template<class Field >
std::ostream & WriteMatrix (
    std::ostream & c,
    const Field & F,
    size_t m,
    size_t n,
    typename Field::ConstElement_ptr A,
    size_t lda,
    FFLAS_FORMAT format,
    bool column_major) [inline]
```

WriteMatrix: write a matrix to an output stream.

**Parameters**

<i>c</i>	output stream
<i>F</i>	base field
<i>m</i>	row dimension
<i>n</i>	column dimension
<i>A</i>	matrix
<i>format</i>	input format (FflasAuto, FflasDense, FflasSMS, FflasBinary)
<i>column_major</i>	whether the matrix is stored in column or row major (row by default)

**11.1.3.326 preamble()**

```
void preamble (
    std::ifstream & ifs,
    FFLAS_FORMAT & format) [inline]
```

**11.1.3.327 ReadMatrix()** [1/2]

```
template<class Field >
Field::Element_ptr ReadMatrix (
    std::ifstream & ifs,
    Field & F,
    size_t & m,
    size_t & n,
    typename Field::Element_ptr & A,
    FFLAS_FORMAT format = FflasAuto)
```

ReadMatrix: read a matrix from an input stream.

**Parameters**

	<i>ifs</i>	input stream
	<i>F</i>	base field
out	<i>m</i>	row dimension
out	<i>n</i>	column dimension
out	<i>A</i>	output matrix
	<i>format</i>	input format (FflasAuto, FflasDense, FflasSMS, FflasBinary)



**11.1.3.328 ReadMatrix()** [2/2]

```
template<class Field >
Field::Element_ptr ReadMatrix (
    const std::string & matrix_file,
    Field & F,
    size_t & m,
    size_t & n,
    typename Field::Element_ptr & A,
    FFLAS_FORMAT format = FflasAuto) [inline]
```

ReadMatrix: read a matrix from a file.

**Parameters**

	<i>matrix_file</i>	filename
	<i>F</i>	base field
out	<i>m</i>	row dimension
out	<i>n</i>	column dimension
out	<i>A</i>	output matrix
	<i>format</i>	input format (FflasAuto, FflasDense, FflasSMS, FflasBinary)

**11.1.3.329 WriteMatrix()** [2/2]

```
template<class Field >
void WriteMatrix (
    std::string & matrix_file,
    const Field & F,
    int m,
    int n,
    typename Field::ConstElement_ptr A,
    size_t lda,
    FFLAS_FORMAT format = FflasDense,
    bool column_major = false)
```

WriteMatrix: write a matrix to a file.

**Parameters**

<i>matrix_file</i>	file name
<i>F</i>	base field
<i>m</i>	row dimension
<i>n</i>	column dimension
<i>A</i>	matrix
<i>format</i>	input format (FflasAuto, FflasDense, FflasSMS, FflasBinary)
<i>column_major</i>	whether the matrix is stored in column or row major (row by default)

**11.1.3.330 WritePermutation()**

```
std::ostream & WritePermutation (
    std::ostream & c,
    const size_t * P,
    size_t N) [inline]
```

WritePermutation: write a permutation matrix to an output stream.

## Parameters

<i>c</i>	output stream
<i>P</i>	permutation
<i>N</i>	size of the permutation

**11.1.3.331 alignable()**

```
template<class Element >
bool alignable () [inline]
```

**11.1.3.332 alignable< Givaro::Integer \* >()**

```
template<>
bool alignable< Givaro::Integer * > () [inline]
```

**11.1.3.333 fflas\_new() [5/7]**

```
template<class Field >
Field::Element_ptr fflas_new (
    const Field & F,
    const size_t m,
    const Alignment align = Alignment::DEFAULT) [inline]
```

**11.1.3.334 fflas\_new() [6/7]**

```
template<class Field >
Field::Element_ptr fflas_new (
    const Field & F,
    const size_t m,
    const size_t n,
    const Alignment align = Alignment::DEFAULT) [inline]
```

**11.1.3.335 fflas\_new() [7/7]**

```
template<class Element >
Element * fflas_new (
    const size_t m,
    const Alignment align = Alignment::DEFAULT) [inline]
```

**11.1.3.336 fflas\_delete() [3/4]**

```
template<class Element_ptr >
void fflas_delete (
    Element_ptr A) [inline]
```

**11.1.3.337 fflas\_delete()** [4/4]

```
template<class Ptr , class ... Args>
void fflas_delete (
    Ptr p,
    Args ... args) [inline]
```

**11.1.3.338 prefetch()**

```
void prefetch (
    const int64_t * ) [inline]
```

**11.1.3.339 getTLBSize()**

```
void getTLBSize (
    int & tlb) [inline]
```

**11.1.3.340 queryCacheSizes()**

```
void queryCacheSizes (
    int & l1,
    int & l2,
    int & l3) [inline]
```

Queries and returns the cache sizes in Bytes of the L1, L2, and L3 data caches respectively

**11.1.3.341 queryL1CacheSize()**

```
int queryL1CacheSize () [inline]
```

**Returns**

the size in Bytes of the L1 data cache

**11.1.3.342 queryTopLevelCacheSize()**

```
int queryTopLevelCacheSize () [inline]
```

**Returns**

the size in Bytes of the L2 or L3 cache if this later is present

**11.1.3.343 getSeed()**

```
uint64_t getSeed ()
```

## 11.2 FFLAS::\_ftranspose\_impl Namespace Reference

### Functions

- `template<size_t bs, typename Field , typename BTSimd >`  
`void not_inplace (const Field &F, const BTSimd &BTS, const size_t m, const size_t n, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb)`
- `template<size_t bs, typename Field , typename BTSimd >`  
`void square_inplace (const Field &F, const BTSimd &BTS, const size_t m, typename Field::Element_ptr A, const size_t lda)`
- `template<size_t bs, typename Field , typename BTSimd >`  
`void nonsquare_inplace_v1 (const Field &F, const BTSimd &BTS, const size_t m, const size_t n, typename Field::Element_ptr A)`
- `template<size_t bs, typename Field , typename BTSimd >`  
`void nonsquare_inplace_v2 (const Field &F, const BTSimd &BTS, const size_t m, const size_t n, typename Field::Element_ptr A)`

### 11.2.1 Function Documentation

#### 11.2.1.1 not\_inplace()

```
template<size_t bs, typename Field , typename BTSimd >
void not_inplace (
    const Field & F,
    const BTSimd & BTS,
    const size_t m,
    const size_t n,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::Element_ptr B,
    const size_t ldb)
```

#### 11.2.1.2 square\_inplace()

```
template<size_t bs, typename Field , typename BTSimd >
void square_inplace (
    const Field & F,
    const BTSimd & BTS,
    const size_t m,
    typename Field::Element_ptr A,
    const size_t lda)
```

#### 11.2.1.3 nonsquare\_inplace\_v1()

```
template<size_t bs, typename Field , typename BTSimd >
void nonsquare_inplace_v1 (
    const Field & F,
    const BTSimd & BTS,
    const size_t m,
    const size_t n,
    typename Field::Element_ptr A)
```

#### 11.2.1.4 nonsquare\_inplace\_v2()

```
template<size_t bs, typename Field , typename BTSimd >
void nonsquare_inplace_v2 (
    const Field & F,
    const BTSimd & BTS,
    const size_t m,
    const size_t n,
    typename Field::Element_ptr A)
```

## 11.3 FFLAS::BLAS3 Namespace Reference

### Functions

- template<class Field >
 void Bini (const Field &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE tb, const size\_t mr, const size\_t nr, const size\_t kr, const typename Field::Element alpha, const typename Field::Element\_ptr A, const size\_t lda, const typename Field::Element\_ptr B, const size\_t ldb, const typename Field::Element beta, typename Field::Element\_ptr C, const size\_t ldc, const size\_t kmax, const size\_t w, const FFLAS\_BASE base, const size\_t rec\_level)
- template<class Field , class FieldTrait , class Strat , class Param >
 Field::Element\_ptr WinoPar (const Field &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE tb, const size\_t mr, const size\_t nr, const size\_t kr, const typename Field::Element alpha, typename Field::ConstElement\_ptr A, const size\_t lda, typename Field::ConstElement\_ptr B, const size\_t ldb, const typename Field::Element beta, typename Field::Element\_ptr C, const size\_t ldc, MMHelper< Field, MMHelperAlgo::WinogradPar, FieldTrait, ParSeqHelper::Parallel< Strat, Param > > &WH)
- template<class Field , class FieldTrait >
 void Winograd (const Field &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE tb, const size\_t mr, const size\_t nr, const size\_t kr, const typename Field::Element alpha, typename Field::ConstElement\_ptr A, const size\_t lda, typename Field::ConstElement\_ptr B, const size\_t ldb, const typename Field::Element beta, typename Field::Element\_ptr C, const size\_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)
- template<class Field , class FieldTrait >
 void WinogradAcc\_3\_23 (const Field &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE tb, const size\_t mr, const size\_t nr, const size\_t kr, const typename Field::Element alpha, typename Field::ConstElement\_ptr A, const size\_t lda, typename Field::ConstElement\_ptr B, const size\_t ldb, const typename Field::Element beta, typename Field::Element\_ptr C, const size\_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)
- template<class Field , class FieldTrait >
 void WinogradAcc\_3\_21 (const Field &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE tb, const size\_t mr, const size\_t nr, const size\_t kr, const typename Field::Element alpha, typename Field::ConstElement\_ptr A, const size\_t lda, typename Field::ConstElement\_ptr B, const size\_t ldb, const typename Field::Element beta, typename Field::Element\_ptr C, const size\_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)
- template<class Field , class FieldTrait >
 void WinogradAcc\_2\_24 (const Field &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE tb, const size\_t mr, const size\_t nr, const size\_t kr, const typename Field::Element alpha, const typename Field::Element\_ptr A, const size\_t lda, const typename Field::Element\_ptr B, const size\_t ldb, const typename Field::Element beta, typename Field::Element\_ptr C, const size\_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)
- template<class Field , class FieldTrait >
 void WinogradAcc\_2\_27 (const Field &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE tb, const size\_t mr, const size\_t nr, const size\_t kr, const typename Field::Element alpha, const typename Field::Element\_ptr A, const size\_t lda, const typename Field::Element\_ptr B, const size\_t ldb, const typename Field::Element beta, typename Field::Element\_ptr C, const size\_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)

- `template<class Field , class FieldTrait >`  
`void WinogradAcc_LR (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const MMHelper< Field, MMHelperAlgo::Winograd, Field↵Trait > &WH)`
- `template<class Field , class FieldTrait >`  
`void WinogradAcc_R_S (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, const typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)`
- `template<class Field , class FieldTrait >`  
`void WinogradAcc_L_S (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, const typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)`
- `template<class Field , class FieldTrait >`  
`void Winograd_LR_S (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const MMHelper< Field, MMHelperAlgo::Winograd, Field↵Trait > &WH)`
- `template<class Field , class FieldTrait >`  
`void Winograd_L_S (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, const typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)`
- `template<class Field , class FieldTrait >`  
`void Winograd_R_S (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, const typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)`

### 11.3.1 Function Documentation

#### 11.3.1.1 Bini()

```
template<class Field >
void Bini (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t mr,
    const size_t nr,
    const size_t kr,
    const typename Field::Element alpha,
    const typename Field::Element_ptr A,
    const size_t lda,
    const typename Field::Element_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
```

```

typename Field::Element_ptr C,
const size_t ldc,
const size_t kmax,
const size_t w,
const FFLAS_BASE base,
const size_t rec_level) [inline]

```

### 11.3.1.2 WinoPar()

```

template<class Field , class FieldTrait , class Strat , class Param >
Field::Element_ptr WinoPar (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t mr,
    const size_t nr,
    const size_t kr,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::WinogradPar, FieldTrait, ParSeqHelper::Parallel<
Strat, Param > > & WH) [inline]

```

### 11.3.1.3 Winograd()

```

template<class Field , class FieldTrait >
void Winograd (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t mr,
    const size_t nr,
    const size_t kr,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]

```

### 11.3.1.4 WinogradAcc\_3\_23()

```

template<class Field , class FieldTrait >
void WinogradAcc_3_23 (
    const Field & F,

```

```

const FFLAS_TRANSPOSE ta,
const FFLAS_TRANSPOSE tb,
const size_t mr,
const size_t nr,
const size_t kr,
const typename Field::Element alpha,
typename Field::ConstElement_ptr A,
const size_t lda,
typename Field::ConstElement_ptr B,
const size_t ldb,
const typename Field::Element beta,
typename Field::Element_ptr C,
const size_t ldc,
MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]

```

#### 11.3.1.5 WinogradAcc\_3\_21()

```

template<class Field , class FieldTrait >
void WinogradAcc_3_21 (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t mr,
    const size_t nr,
    const size_t kr,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]

```

#### 11.3.1.6 WinogradAcc\_2\_24()

```

template<class Field , class FieldTrait >
void WinogradAcc_2_24 (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t mr,
    const size_t nr,
    const size_t kr,
    const typename Field::Element alpha,
    const typename Field::Element_ptr A,
    const size_t lda,
    const typename Field::Element_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]

```



### 11.3.1.7 WinogradAcc\_2\_27()

```

template<class Field , class FieldTrait >
void WinogradAcc_2_27 (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t mr,
    const size_t nr,
    const size_t kr,
    const typename Field::Element alpha,
    const typename Field::Element_ptr A,
    const size_t lda,
    const typename Field::Element_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]

```

### 11.3.1.8 WinogradAcc\_LR()

```

template<class Field , class FieldTrait >
void WinogradAcc_LR (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t mr,
    const size_t nr,
    const size_t kr,
    const typename Field::Element alpha,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]

```

### 11.3.1.9 WinogradAcc\_R\_S()

```

template<class Field , class FieldTrait >
void WinogradAcc_R_S (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t mr,
    const size_t nr,
    const size_t kr,
    const typename Field::Element alpha,
    const typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr B,

```

```

const size_t ldb,
const typename Field::Element beta,
typename Field::Element_ptr C,
const size_t ldc,
const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]

```

#### 11.3.1.10 WinogradAcc\_L\_S()

```

template<class Field , class FieldTrait >
void WinogradAcc_L_S (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t mr,
    const size_t nr,
    const size_t kr,
    const typename Field::Element alpha,
    typename Field::Element_ptr A,
    const size_t lda,
    const typename Field::Element_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]

```

#### 11.3.1.11 Winograd\_LR\_S()

```

template<class Field , class FieldTrait >
void Winograd_LR_S (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t mr,
    const size_t nr,
    const size_t kr,
    const typename Field::Element alpha,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]

```

#### 11.3.1.12 Winograd\_L\_S()

```

template<class Field , class FieldTrait >
void Winograd_L_S (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,

```

```

const size_t mr,
const size_t nr,
const size_t kr,
const typename Field::Element alpha,
typename Field::Element_ptr A,
const size_t lda,
const typename Field::Element_ptr B,
const size_t ldb,
const typename Field::Element beta,
typename Field::Element_ptr C,
const size_t ldc,
const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]

```

### 11.3.1.13 Winograd\_R\_S()

```

template<class Field , class FieldTrait >
void Winograd_R_S (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t mr,
    const size_t nr,
    const size_t kr,
    const typename Field::Element alpha,
    const typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]

```

## 11.4 FFLAS::csr\_hyb\_details Namespace Reference

### Data Structures

- struct [Coo](#)
- struct [Info](#)

## 11.5 FFLAS::CuttingStrategy Namespace Reference

### Data Structures

- struct [Block](#)
- struct [Column](#)
- struct [Recursive](#)
- struct [Row](#)
- struct [Single](#)

## Typedefs

- typedef [Row](#) [RNSModulus](#)

## 11.5.1 Typedef Documentation

### 11.5.1.1 RNSModulus

typedef [Row](#) [RNSModulus](#)

## 11.6 FFLAS::details Namespace Reference

### Functions

- template<class [Field](#) , bool ADD>  
std::enable\_if< [FFLAS::support\\_simd\\_add](#)< typename [Field::Element](#) >::value, void >::type [fadd](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t inca, typename [Field::ConstElement\\_ptr](#) B, const size\_t incb, typename [Field::Element\\_ptr](#) C, const size\_t incc, [FieldCategories::ModularTag](#))
- template<class [Field](#) , bool ADD>  
std::enable\_if<![FFLAS::support\\_simd\\_add](#)< typename [Field::Element](#) >::value, void >::type [fadd](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t inca, typename [Field::ConstElement\\_ptr](#) B, const size\_t incb, typename [Field::Element\\_ptr](#) C, const size\_t incc, [FieldCategories::ModularTag](#))
- template<class [Field](#) , bool ADD>  
void [fadd](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t inca, typename [Field::ConstElement\\_ptr](#) B, const size\_t incb, typename [Field::Element\\_ptr](#) C, const size\_t incc, [FieldCategories::GenericTag](#))
- template<class [Field](#) , bool ADD>  
std::enable\_if<![FFLAS::support\\_simd\\_add](#)< typename [Field::Element](#) >::value, void >::type [fadd](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t inca, typename [Field::ConstElement\\_ptr](#) B, const size\_t incb, typename [Field::Element\\_ptr](#) C, const size\_t incc, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) , bool ADD>  
std::enable\_if< [FFLAS::support\\_simd\\_add](#)< typename [Field::Element](#) >::value, void >::type [fadd](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t inca, typename [Field::ConstElement\\_ptr](#) B, const size\_t incb, typename [Field::Element\\_ptr](#) C, const size\_t incc, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
std::enable\_if< [FFLAS::support\\_fast\\_mod](#)< typename [Field::Element](#) >::value, void >::type [faxpy](#) (const [Field](#) &F, const size\_t N, const typename [Field::Element](#) a, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, typename [Field::Element\\_ptr](#) Y, const size\_t incY, [FieldCategories::ModularTag](#))
- template<class [Field](#) , class FC >  
void [faxpy](#) (const [Field](#) &F, const size\_t N, const typename [Field::Element](#) a, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, typename [Field::Element\\_ptr](#) Y, const size\_t incY, FC)
- template<class [Field](#) >  
std::enable\_if< [FFLAS::support\\_fast\\_mod](#)< typename [Field::Element](#) >::value, void >::type [freduce](#) (const [Field](#) &F, const size\_t m, typename [Field::Element\\_ptr](#) A, const size\_t incX, [FieldCategories::ModularTag](#))
- template<class [Field](#) >  
std::enable\_if< [FFLAS::support\\_fast\\_mod](#)< typename [Field::Element](#) >::value, void >::type [freduce](#) (const [Field](#) &F, const size\_t m, typename [Field::ConstElement\\_ptr](#) B, const size\_t incY, typename [Field::Element\\_ptr](#) A, const size\_t incX, [FieldCategories::ModularTag](#))

- template<class [Field](#) , class FC >  
void [freduce](#) (const [Field](#) &F, const size\_t m, typename [Field::Element\\_ptr](#) A, const size\_t incX, FC)
- template<class [Field](#) , class FC >  
void [freduce](#) (const [Field](#) &F, const size\_t m, typename [Field::ConstElement\\_ptr](#) B, const size\_t incY, typename [Field::Element\\_ptr](#) A, const size\_t incX, FC)
- template<class [Field](#) >  
std::enable\_if< [FFLAS::support\\_fast\\_mod](#)< typename [Field::Element](#) >::value, void >::type [fscalin](#) (const [Field](#) &F, const size\_t N, const typename [Field::Element](#) a, typename [Field::Element\\_ptr](#) X, const size\_t incX, [FieldCategories::ModularTag](#))
- template<class [Field](#) >  
std::enable\_if< [FFLAS::support\\_fast\\_mod](#)< typename [Field::Element](#) >::value, void >::type [fscal](#) (const [Field](#) &F, const size\_t N, const typename [Field::Element](#) a, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, typename [Field::Element\\_ptr](#) Y, const size\_t incY, [FieldCategories::ModularTag](#))
- template<class [Field](#) , class FC >  
void [fscalin](#) (const [Field](#) &F, const size\_t n, const typename [Field::Element](#) a, typename [Field::Element\\_ptr](#) X, const size\_t incX, FC)
- template<class [Field](#) , class FC >  
void [fscal](#) (const [Field](#) &F, const size\_t N, const typename [Field::Element](#) a, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, typename [Field::Element\\_ptr](#) Y, const size\_t incY, FC)
- template<enum [number\\_kind](#) K>  
void [igebb44](#) (size\_t i, size\_t j, size\_t depth, size\_t pdeth, const int64\_t alpha, const int64\_t \*blA, const int64\_t \*blB, int64\_t \*C, size\_t ldc)
- template<enum [number\\_kind](#) K>  
void [igebb24](#) (size\_t i, size\_t j, size\_t depth, size\_t pdeth, const int64\_t alpha, const int64\_t \*blA, const int64\_t \*blB, int64\_t \*C, size\_t ldc)
- template<enum [number\\_kind](#) K>  
void [igebb14](#) (size\_t i, size\_t j, size\_t depth, size\_t pdeth, const int64\_t alpha, const int64\_t \*blA, const int64\_t \*blB, int64\_t \*C, size\_t ldc)
- template<enum [number\\_kind](#) K>  
void [igebb41](#) (size\_t i, size\_t j, size\_t depth, size\_t pdeth, const int64\_t alpha, const int64\_t \*blA, const int64\_t \*blB, int64\_t \*C, size\_t ldc)
- template<enum [number\\_kind](#) K>  
void [igebb21](#) (size\_t i, size\_t j, size\_t depth, size\_t pdeth, const int64\_t alpha, const int64\_t \*blA, const int64\_t \*blB, int64\_t \*C, size\_t ldc)
- template<enum [number\\_kind](#) K>  
void [igebb11](#) (size\_t i, size\_t j, size\_t depth, size\_t pdeth, const int64\_t alpha, const int64\_t \*blA, const int64\_t \*blB, int64\_t \*C, size\_t ldc)
- template<enum [number\\_kind](#) K>  
void [igebp](#) (size\_t rows, size\_t cols, size\_t depth, const int64\_t alpha, const int64\_t \*blockA, size\_t lda, const int64\_t \*blockB, size\_t ldb, int64\_t \*C, size\_t ldc)
- template<size\_t k, bool transpose>  
void [pack\\_lhs](#) (int64\_t \*XX, const int64\_t \*X, size\_t ldx, size\_t rows, size\_t cols)
- template<size\_t k, bool transpose>  
void [pack\\_rhs](#) (int64\_t \*XX, const int64\_t \*X, size\_t ldx, size\_t rows, size\_t cols)
- void [gebp](#) (size\_t rows, size\_t cols, size\_t depth, int64\_t \*C, size\_t ldc, const int64\_t \*blockA, size\_t lda, const int64\_t \*BlockB, size\_t ldb, int64\_t \*BlockW)
- void [BlockingFactor](#) (size\_t &m, size\_t &n, size\_t &k)

## 11.6.1 Function Documentation

### 11.6.1.1 fadd() [1/5]

```
template<class Field , bool ADD>
std::enable_if< FFLAS::support\_simd\_add< typename Field::Element >::value, void >::type fadd (
    const Field & F,
```

```

    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t inca,
    typename Field::ConstElement_ptr B,
    const size_t incb,
    typename Field::Element_ptr C,
    const size_t incc,
    FieldCategories::ModularTag )

```

#### 11.6.1.2 fadd() [2/5]

```

template<class Field , bool ADD>
std::enable_if<!FFLAS::support_simd_add< typenameField::Element >::value, void >::type fadd (
    const Field & F,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t inca,
    typename Field::ConstElement_ptr B,
    const size_t incb,
    typename Field::Element_ptr C,
    const size_t incc,
    FieldCategories::ModularTag )

```

#### 11.6.1.3 fadd() [3/5]

```

template<class Field , bool ADD>
void fadd (
    const Field & F,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t inca,
    typename Field::ConstElement_ptr B,
    const size_t incb,
    typename Field::Element_ptr C,
    const size_t incc,
    FieldCategories::GenericTag )

```

#### 11.6.1.4 fadd() [4/5]

```

template<class Field , bool ADD>
std::enable_if<!FFLAS::support_simd_add< typenameField::Element >::value, void >::type fadd (
    const Field & F,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t inca,
    typename Field::ConstElement_ptr B,
    const size_t incb,
    typename Field::Element_ptr C,
    const size_t incc,
    FieldCategories::UnparametricTag ) [inline]

```

**11.6.1.5 fadd()** [5/5]

```
template<class Field , bool ADD>
std::enable_if< FFLAS::support_simd_add< typenameField::Element >::value, void >::type fadd (
    const Field & F,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t inca,
    typename Field::ConstElement_ptr B,
    const size_t incb,
    typename Field::Element_ptr C,
    const size_t incc,
    FieldCategories::UnparametricTag ) [inline]
```

**11.6.1.6 faxpy()** [1/2]

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type faxpy
(
    const Field & F,
    const size_t N,
    const typename Field::Element a,
    typename Field::ConstElement_ptr X,
    const size_t incX,
    typename Field::Element_ptr Y,
    const size_t incY,
    FieldCategories::ModularTag ) [inline]
```

**11.6.1.7 faxpy()** [2/2]

```
template<class Field , class FC >
void faxpy (
    const Field & F,
    const size_t N,
    const typename Field::Element a,
    typename Field::ConstElement_ptr X,
    const size_t incX,
    typename Field::Element_ptr Y,
    const size_t incY,
    FC ) [inline]
```

**11.6.1.8 freduce()** [1/4]

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type freduce
(
    const Field & F,
    const size_t m,
    typename Field::Element_ptr A,
    const size_t incX,
    FieldCategories::ModularTag ) [inline]
```

**11.6.1.9 freduce()** [2/4]

```

template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type freduce
(
    const Field & F,
    const size_t m,
    typename Field::ConstElement_ptr B,
    const size_t incY,
    typename Field::Element_ptr A,
    const size_t incX,
    FieldCategories::ModularTag ) [inline]

```

**11.6.1.10 freduce()** [3/4]

```

template<class Field , class FC >
void freduce (
    const Field & F,
    const size_t m,
    typename Field::Element_ptr A,
    const size_t incX,
    FC ) [inline]

```

**11.6.1.11 freduce()** [4/4]

```

template<class Field , class FC >
void freduce (
    const Field & F,
    const size_t m,
    typename Field::ConstElement_ptr B,
    const size_t incY,
    typename Field::Element_ptr A,
    const size_t incX,
    FC ) [inline]

```

**11.6.1.12 fscaln()** [1/2]

```

template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type fscaln
(
    const Field & F,
    const size_t N,
    const typename Field::Element a,
    typename Field::Element_ptr X,
    const size_t incX,
    FieldCategories::ModularTag ) [inline]

```



**11.6.1.13 fscal()** [1/2]

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type fscal
(
    const Field & F,
    const size_t N,
    const typename Field::Element a,
    typename Field::ConstElement_ptr X,
    const size_t incX,
    typename Field::Element_ptr Y,
    const size_t incY,
    FieldCategories::ModularTag ) [inline]
```

**11.6.1.14 fscaln()** [2/2]

```
template<class Field , class FC >
void fscaln (
    const Field & F,
    const size_t n,
    const typename Field::Element a,
    typename Field::Element_ptr X,
    const size_t incX,
    FC ) [inline]
```

**11.6.1.15 fscal()** [2/2]

```
template<class Field , class FC >
void fscal (
    const Field & F,
    const size_t N,
    const typename Field::Element a,
    typename Field::ConstElement_ptr X,
    const size_t incX,
    typename Field::Element_ptr Y,
    const size_t incY,
    FC ) [inline]
```

**11.6.1.16 igebb44()**

```
template<enum number_kind K>
void igebb44 (
    size_t i,
    size_t j,
    size_t depth,
    size_t pdeth,
    const int64_t alpha,
    const int64_t * blA,
    const int64_t * blB,
    int64_t * C,
    size_t ldc) [inline]
```

**11.6.1.17 igebb24()**

```
template<enum number_kind K>
void igebb24 (
    size_t i,
    size_t j,
    size_t depth,
    size_t pdeth,
    const int64_t alpha,
    const int64_t * blA,
    const int64_t * blB,
    int64_t * C,
    size_t ldc) [inline]
```

**11.6.1.18 igebb14()**

```
template<enum number_kind K>
void igebb14 (
    size_t i,
    size_t j,
    size_t depth,
    size_t pdeth,
    const int64_t alpha,
    const int64_t * blA,
    const int64_t * blB,
    int64_t * C,
    size_t ldc) [inline]
```

**11.6.1.19 igebb41()**

```
template<enum number_kind K>
void igebb41 (
    size_t i,
    size_t j,
    size_t depth,
    size_t pdeth,
    const int64_t alpha,
    const int64_t * blA,
    const int64_t * blB,
    int64_t * C,
    size_t ldc) [inline]
```

bug ,B\_0 dans VEC\_MADD\_32 ?

bug ,B\_0 dans VEC\_MADD\_32 ?

**11.6.1.20 igebb21()**

```
template<enum number_kind K>
void igebb21 (
    size_t i,
    size_t j,
    size_t depth,
```

```

    size_t pdeth,
    const int64_t alpha,
    const int64_t * blA,
    const int64_t * blB,
    int64_t * C,
    size_t ldc) [inline]

```

#### 11.6.1.21 igebb11()

```

template<enum number_kind K>
void igebb11 (
    size_t i,
    size_t j,
    size_t depth,
    size_t pdeth,
    const int64_t alpha,
    const int64_t * blA,
    const int64_t * blB,
    int64_t * C,
    size_t ldc) [inline]

```

#### 11.6.1.22 igebp()

```

template<enum number_kind K>
void igebp (
    size_t rows,
    size_t cols,
    size_t depth,
    const int64_t alpha,
    const int64_t * blockA,
    size_t lda,
    const int64_t * blockB,
    size_t ldb,
    int64_t * C,
    size_t ldc)

```

#### 11.6.1.23 pack\_lhs()

```

template<size_t k, bool transpose>
void pack_lhs (
    int64_t * XX,
    const int64_t * X,
    size_t ldx,
    size_t rows,
    size_t cols)

```

**Bug** this is fassign

**Bug** this is fassign

**Bug** this is fassign

**Bug** this is fassign

### 11.6.1.24 pack\_rhs()

```
template<size_t k, bool transpose>
void pack_rhs (
    int64_t * XX,
    const int64_t * X,
    size_t ldx,
    size_t rows,
    size_t cols)
```

**Bug** this is fassign

**Bug** this is fassign

**Bug** this is fassign

**Bug** this is fassign

### 11.6.1.25 gebp()

```
void gebp (
    size_t rows,
    size_t cols,
    size_t depth,
    int64_t * C,
    size_t ldc,
    const int64_t * blockA,
    size_t lda,
    const int64_t * BlockB,
    size_t ldb,
    int64_t * BlockW)
```

### 11.6.1.26 BlockingFactor()

```
void BlockingFactor (
    size_t & m,
    size_t & n,
    size_t & k) [inline]
```

## 11.7 FFLAS::details\_spmv Namespace Reference

### Data Structures

- struct [Coo](#)

## 11.8 FFLAS::ElementCategories Namespace Reference

### Data Structures

- struct [ArbitraryPrecIntTag](#)  
*Arbitrary precision integers: GMP.*
- struct [FixedPrecIntTag](#)  
*Fixed precision integers above machine precision: Givaro::reclnt.*
- struct [GenericTag](#)  
*default is generic*
- struct [MachineFloatTag](#)  
*float or double*
- struct [MachineIntTag](#)  
*short, int, long, long long, and unsigned variants*
- struct [RNSElementTag](#)  
*Representation in a Residue Number System.*

## 11.9 FFLAS::FieldCategories Namespace Reference

Traits and categories will need to be placed in a proper file later.

### Data Structures

- struct [GenericTag](#)  
*generic ring.*
- struct [ModularTag](#)  
*This is a modular field like e.g. `Modular<T>` or `ModularBalanced<T>`*
- struct [UnparametricTag](#)  
*If the field uses a representation with infix operators.*

### 11.9.1 Detailed Description

Traits and categories will need to be placed in a proper file later.

## 11.10 FFLAS::MMHelperAlgo Namespace Reference

### Data Structures

- struct [Auto](#)
- struct [Bini](#)
- struct [Classic](#)
- struct [DivideAndConquer](#)
- struct [Winograd](#)
- struct [WinogradPar](#)

## 11.11 FFLAS::ModeCategories Namespace Reference

Specifies the mode of action for an algorithm w.r.t.

### Data Structures

- struct [ConvertTo](#)  
*Force conversion to appropriate element type of ElementCategory T.*
- struct [DefaultBoundedTag](#)  
*Use standard field operations, but keeps track of bounds on input and output.*
- struct [DefaultTag](#)  
*No specific mode of action: use standard field operations.*
- struct [DelayedTag](#)  
*Performs field operations with delayed mod reductions. Ensures result is reduced.*
- struct [LazyTag](#)  
*Performs field operations with delayed mod only when necessary. Result may not be reduced.*

### 11.11.1 Detailed Description

Specifies the mode of action for an algorithm w.r.t.

its field

## 11.12 FFLAS::ParSeqHelper Namespace Reference

[ParSeqHelper](#) for both fgemm and ftrsm.

### Data Structures

- struct [Compose](#)
- struct [Parallel](#)
- struct [Sequential](#)

### 11.12.1 Detailed Description

[ParSeqHelper](#) for both fgemm and ftrsm.

[ParSeqHelper](#) for both fgemm and ftrsm

## 11.13 FFLAS::Protected Namespace Reference

### Data Structures

- class [AreEqual](#)
- class [AreEqual< X, X >](#)
- class [ftrmmLeftLowerNoTransNonUnit](#)
- class [ftrmmLeftLowerNoTransUnit](#)
- class [ftrmmLeftLowerTransNonUnit](#)
- class [ftrmmLeftLowerTransUnit](#)
- class [ftrmmLeftUpperNoTransNonUnit](#)
- class [ftrmmLeftUpperNoTransUnit](#)
- class [ftrmmLeftUpperTransNonUnit](#)
- class [ftrmmLeftUpperTransUnit](#)
- class [ftrmmRightLowerNoTransNonUnit](#)
- class [ftrmmRightLowerNoTransUnit](#)
- class [ftrmmRightLowerTransNonUnit](#)
- class [ftrmmRightLowerTransUnit](#)
- class [ftrmmRightUpperNoTransNonUnit](#)
- class [ftrmmRightUpperNoTransUnit](#)
- class [ftrmmRightUpperTransNonUnit](#)
- class [ftrmmRightUpperTransUnit](#)
- class [ftrsmLeftLowerNoTransNonUnit](#)
- class [ftrsmLeftLowerNoTransUnit](#)
- class [ftrsmLeftLowerTransNonUnit](#)
- class [ftrsmLeftLowerTransUnit](#)
- class [ftrsmLeftUpperNoTransNonUnit](#)

*Computes the maximal size for delaying the modular reduction in a triangular system resolution.*

- class [ftrsmLeftUpperNoTransUnit](#)
- class [ftrsmLeftUpperTransNonUnit](#)
- class [ftrsmLeftUpperTransUnit](#)
- class [ftrsmRightLowerNoTransNonUnit](#)
- class [ftrsmRightLowerNoTransUnit](#)
- class [ftrsmRightLowerTransNonUnit](#)
- class [ftrsmRightLowerTransUnit](#)
- class [ftrsmRightUpperNoTransNonUnit](#)
- class [ftrsmRightUpperNoTransUnit](#)
- class [ftrsmRightUpperTransNonUnit](#)
- class [ftrsmRightUpperTransUnit](#)

### Functions

- template<class [Field](#) >  
double [computeFactorClassic](#) (const [Field](#) &F)
- template<> double [computeFactorClassic](#) (const [Givaro::ModularBalanced](#)< double > &F)
- template<> double [computeFactorClassic](#) (const [Givaro::ModularBalanced](#)< float > &F)
- template<class [Field](#) >  
size\_t [DotProdBoundClassic](#) (const [Field](#) &F, const typename [Field::Element](#) &beta)
- template<class [Field](#) >  
size\_t [TRSMBound](#) (const [Field](#) &)  
  
*TRSMBound.*
- template<class [Element](#) >  
size\_t [TRSMBound](#) (const [Givaro::Modular](#)< [Element](#) > &F)

*Specialization for positive modular representation over double Computes nmax s.t.*

- `template<class Element >`  
`size_t TRSMBound (const Givaro::ModularBalanced< Element > &F)`

*Specialization for balanced modular representation over double.*

- `template<class NewField , class Field , class FieldMode >`  
`Field::Element_ptr fgemm_convert (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, FieldMode > &H)`
- `template<class Field , class Element , class AlgoT , class ParSeqTrait >`  
`bool NeedPreAddReduction (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, Element &Op2min, Element &Op2max, MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > &WH)`
- `template<class Field , class Element , class AlgoT , class ModeT , class ParSeqTrait >`  
`bool NeedPreAddReduction (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, Element &Op2min, Element &Op2max, MMHelper< Field, AlgoT, ModeT, ParSeqTrait > &WH)`
- `template<class Field , class Element , class AlgoT , class ParSeqTrait >`  
`bool NeedPreSubReduction (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, Element &Op2min, Element &Op2max, MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > &WH)`
- `template<class Field , class Element , class AlgoT , class ModeT , class ParSeqTrait >`  
`bool NeedPreSubReduction (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, Element &Op2min, Element &Op2max, MMHelper< Field, AlgoT, ModeT, ParSeqTrait > &WH)`
- `template<class Field , class Element , class AlgoT , class ParSeqTrait >`  
`bool NeedDoublePreAddReduction (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, Element &Op2min, Element &Op2max, Element beta, MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > &WH)`
- `template<class Field , class Element , class AlgoT , class ModeT , class ParSeqTrait >`  
`bool NeedDoublePreAddReduction (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, Element &Op2min, Element &Op2max, Element beta, MMHelper< Field, AlgoT, ModeT, ParSeqTrait > &WH)`
- `template<class Field , class AlgoT , class ParSeqTrait >`  
`void ScalAndReduce (const Field &F, const size_t N, const typename Field::Element alpha, typename Field::Element_ptr X, const size_t incX, const MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > &H)`
- `template<class Field , class AlgoT , class ParSeqTrait >`  
`void ScalAndReduce (const Field &F, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, const MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > &H)`
- `template<class Field >`  
`Field::Element_ptr fsquareCommon (const Field &F, const FFLAS_TRANSPOSE ta, const size_t n, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc)`
- `template<class Field >`  
`int WinogradThreshold (const Field &F)`  
*Computes the number of recursive levels to perform.*
- `template<> int WinogradThreshold (const Givaro::Modular< float > &F)`
- `template<> int WinogradThreshold (const Givaro::ModularBalanced< double > &F)`
- `template<> int WinogradThreshold (const Givaro::ModularBalanced< float > &F)`
- `template<class Field >`  
`int WinogradSteps (const Field &F, const size_t &m)`  
*Computes the number of recursive levels to perform.*
- `template<class Field , class FieldMode >`  
`void DynamicPeeling (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda,`



- typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), [FieldMode](#) > &H, const typename [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), [FieldMode](#) >::DelayedField::Element Cmin, const typename [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), [FieldMode](#) >::DelayedField::Element Cmax)
- template<class [Field](#) , class [FieldMode](#) >  
void [DynamicPeeling2](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t m, const size\_t n, const size\_t k, const size\_t mr, const size\_t nr, const size\_t kr, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), [FieldMode](#) > &H, const typename [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), [FieldMode](#) >::DelayedField::Element Cmin, const typename [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), [FieldMode](#) >::DelayedField::Element Cmax)
  - template<class [Field](#) , class [FieldMode](#) >  
void [WinogradCalc](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t mr, const size\_t nr, const size\_t kr, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), [FieldMode](#) > &H)
  - template<typename [FloatElement](#) , class [Field](#) >  
[Field::Element\\_ptr](#) [fgemv\\_convert](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) Y, const size\_t incY)
  - template<class [FloatElement](#) , class [Field](#) >  
void [fger\\_convert](#) (const [Field](#) &F, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) x, const size\_t incx, typename [Field::ConstElement\\_ptr](#) y, const size\_t incy, typename [Field::Element\\_ptr](#) A, const size\_t lda)
  - template<class [NewField](#) , class [Field](#) , class [FieldMode](#) >  
[Field::Element\\_ptr](#) [fsyrk\\_convert](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) UpLo, const [FFLAS\\_TRANSPOSE](#) trans, const size\_t N, const size\_t K, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, [MMHelper](#)< [Field](#), [MMHelperAlgo::Classic](#), [FieldMode](#) > &H)
  - template<class [Field](#) , class [AlgoT](#) , class [ParSeqTrait](#) >  
void [ScaLAndReduce](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) UpLo, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) A, const size\_t lda, const [MMHelper](#)< [Field](#), [AlgoT](#), [ModeCategories::LazyTag](#), [ParSeqTrait](#) > &H)
  - template<class [Field](#) , class [Element](#) , class [AlgoT](#) , class [ParSeqTrait](#) >  
bool [NeedPreScaLReduction](#) ([Element](#) &Outmin, [Element](#) &Outmax, [Element](#) &Op1min, [Element](#) &Op1max, const [Element](#) &x, [MMHelper](#)< [Field](#), [AlgoT](#), [ModeCategories::LazyTag](#), [ParSeqTrait](#) > &WH)
  - template<class [Field](#) , class [Element](#) , class [AlgoT](#) , class [ModeT](#) , class [ParSeqTrait](#) >  
bool [NeedPreScaLReduction](#) ([Element](#) &Outmin, [Element](#) &Outmax, [Element](#) &Op1min, [Element](#) &Op1max, const [Element](#) &x, [MMHelper](#)< [Field](#), [AlgoT](#), [ModeT](#), [ParSeqTrait](#) > &WH)
  - template<class [Field](#) , class [Element](#) , class [AlgoT](#) , class [ParSeqTrait](#) >  
bool [NeedPreAxyReduction](#) ([Element](#) &Outmin, [Element](#) &Outmax, [Element](#) &Op1min, [Element](#) &Op1max, [Element](#) &Op2min, [Element](#) &Op2max, const [Element](#) &x, [MMHelper](#)< [Field](#), [AlgoT](#), [ModeCategories::LazyTag](#), [ParSeqTrait](#) > &WH)
  - template<class [Field](#) , class [Element](#) , class [AlgoT](#) , class [ModeT](#) , class [ParSeqTrait](#) >  
bool [NeedPreAxyReduction](#) ([Element](#) &Outmin, [Element](#) &Outmax, [Element](#) &Op1min, [Element](#) &Op1max, [Element](#) &Op2min, [Element](#) &Op2max, const [Element](#) &x, [MMHelper](#)< [Field](#), [AlgoT](#), [ModeT](#), [ParSeqTrait](#) > &WH)
  - template<class [DFE](#) >  
size\_t [min\\_types](#) (const [DFE](#) &k)
  - template<> size\_t [min\\_types](#) (const [Reclnt::rint](#)< 6 > &k)
  - template<> size\_t [min\\_types](#) (const [Reclnt::rint](#)< 7 > &k)
  - template<> size\_t [min\\_types](#) (const [Reclnt::rint](#)< 8 > &k)
  - template<> size\_t [min\\_types](#) (const [Reclnt::rint](#)< 9 > &k)
  - template<> size\_t [min\\_types](#) (const [Reclnt::rint](#)< 10 > &k)

- `template<> size_t min_types (const Givaro::Integer &k)`
- `template<class T >`  
`bool unfit (T x)`
- `template<> bool unfit (int64_t x)`
- `template<size_t K>`  
`bool unfit (Reclnt::rint< K > x)`
- `template<> bool unfit (Reclnt::rint< 6 > x)`
- `template<enum FFLAS_TRANSPOSE tA, enum FFLAS_TRANSPOSE tB>`  
`void igemm_colmajor (size_t rows, size_t cols, size_t depth, const int64_t alpha, const int64_t *A, size_t lda, const int64_t *B, size_t ldb, int64_t *C, size_t ldc)`
- `template<enum FFLAS_TRANSPOSE tA, enum FFLAS_TRANSPOSE tB, enum number_kind alpha_kind>`  
`void igemm_colmajor (size_t rows, size_t cols, size_t depth, const int64_t alpha, const int64_t *A, size_t lda, const int64_t *B, size_t ldb, int64_t *C, size_t ldc)`
- `void igemm (const enum FFLAS_TRANSPOSE TransA, const enum FFLAS_TRANSPOSE TransB, size_t rows, size_t cols, size_t depth, const int64_t alpha, const int64_t *A, size_t lda, const int64_t *B, size_t ldb, const int64_t beta, int64_t *C, size_t ldc)`
- `template<class Field >`  
`void MatF2MatD_Triangular (const Field &F, Givaro::DoubleDomain::Element_ptr S, const size_t lds, typename Field::ConstElement_ptr const E, const size_t lde, const size_t m, const size_t n)`
- `template<class Field >`  
`void MatF2MatFI_Triangular (const Field &F, Givaro::FloatDomain::Element_ptr S, const size_t lds, typename Field::ConstElement_ptr const E, const size_t lde, const size_t m, const size_t n)`

## 11.13.1 Function Documentation

### 11.13.1.1 computeFactorClassic() [1/3]

```
template<class Field >
double computeFactorClassic (
    const Field & F) [inline]
```

### 11.13.1.2 computeFactorClassic() [2/3]

```
template<>
double computeFactorClassic (
    const Givaro::ModularBalanced< double > & F) [inline]
```

### 11.13.1.3 computeFactorClassic() [3/3]

```
template<>
double computeFactorClassic (
    const Givaro::ModularBalanced< float > & F) [inline]
```

### 11.13.1.4 DotProdBoundClassic()

```
template<class Field >
size_t DotProdBoundClassic (
    const Field & F,
    const typename Field::Element & beta) [inline]
```

**11.13.1.5 TRSMBound()** [1/3]

```
template<class Field >
size_t TRSMBound (
    const Field & ) [inline]
```

TRSMBound.

computes the maximal size for delaying the modular reduction in a triangular system resolution

This is the default version over an arbitrary field. It is currently never used (the recursive algorithm is run until  $n=1$  in this case)

**Parameters**

$F$	Finite Field/Ring of the computation
-----	--------------------------------------

**11.13.1.6 TRSMBound()** [2/3]

```
template<class Element >
size_t TRSMBound (
    const Givaro::Modular< Element > & F) [inline]
```

Specialization for positive modular representation over double Computes  $n_{\max}$  s.t.

$(p-1)/2 * (p^{\{n_{\max}-1\}} + (p-2)^{\{n_{\max}-1\}}) < 2^{53}$  See [Dumas Giorgi Pernet 06, arXiv:cs/0601133] Specialization for positive modular representation over float. Computes  $n_{\max}$  s.t.  $(p-1)/2 * (p^{\{n_{\max}-1\}} + (p-2)^{\{n_{\max}-1\}}) < 2^{24}$  @pbi See [Dumas Giorgi Pernet 06, arXiv:cs/0601133]

**11.13.1.7 TRSMBound()** [3/3]

```
template<class Element >
size_t TRSMBound (
    const Givaro::ModularBalanced< Element > & F) [inline]
```

Specialization for balanced modular representation over double.

Computes  $n_{\max}$  s.t.  $(p-1)/2 * (((p+1)/2)^{\{n_{\max}-1\}}) < 2^{53}$

**Bibliography** • Dumas Giorgi Pernet 06, arXiv:cs/0601133

**11.13.1.8 fgemmm\_convert()**

```
template<class NewField , class Field , class FieldMode >
Field::Element_ptr fgemmm_convert (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::Winograd, FieldMode > & H) [inline]
```

**11.13.1.9 NeedPreAddReduction() [1/2]**

```
template<class Field , class Element , class AlgoT , class ParSeqTrait >
bool NeedPreAddReduction (
    Element & Outmin,
    Element & Outmax,
    Element & Op1min,
    Element & Op1max,
    Element & Op2min,
    Element & Op2max,
    MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > & WH) [inline]
```

**11.13.1.10 NeedPreAddReduction() [2/2]**

```
template<class Field , class Element , class AlgoT , class ModeT , class ParSeqTrait >
bool NeedPreAddReduction (
    Element & Outmin,
    Element & Outmax,
    Element & Op1min,
    Element & Op1max,
    Element & Op2min,
    Element & Op2max,
    MMHelper< Field, AlgoT, ModeT, ParSeqTrait > & WH) [inline]
```

**11.13.1.11 NeedPreSubReduction() [1/2]**

```
template<class Field , class Element , class AlgoT , class ParSeqTrait >
bool NeedPreSubReduction (
    Element & Outmin,
    Element & Outmax,
    Element & Op1min,
    Element & Op1max,
    Element & Op2min,
    Element & Op2max,
    MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > & WH) [inline]
```

**11.13.1.12 NeedPreSubReduction() [2/2]**

```
template<class Field , class Element , class AlgoT , class ModeT , class ParSeqTrait >
bool NeedPreSubReduction (
    Element & Outmin,
    Element & Outmax,
    Element & Op1min,
    Element & Op1max,
    Element & Op2min,
    Element & Op2max,
    MMHelper< Field, AlgoT, ModeT, ParSeqTrait > & WH) [inline]
```

**11.13.1.13 NeedDoublePreAddReduction() [1/2]**

```
template<class Field , class Element , class AlgoT , class ParSeqTrait >
bool NeedDoublePreAddReduction (
    Element & Outmin,
    Element & Outmax,
    Element & Op1min,
    Element & Op1max,
    Element & Op2min,
    Element & Op2max,
    Element beta,
    MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > & WH) [inline]
```

**11.13.1.14 NeedDoublePreAddReduction() [2/2]**

```
template<class Field , class Element , class AlgoT , class ModeT , class ParSeqTrait >
bool NeedDoublePreAddReduction (
    Element & Outmin,
    Element & Outmax,
    Element & Op1min,
    Element & Op1max,
    Element & Op2min,
    Element & Op2max,
    Element beta,
    MMHelper< Field, AlgoT, ModeT, ParSeqTrait > & WH) [inline]
```

**11.13.1.15 ScalAndReduce() [1/3]**

```
template<class Field , class AlgoT , class ParSeqTrait >
void ScalAndReduce (
    const Field & F,
    const size_t N,
    const typename Field::Element alpha,
    typename Field::Element_ptr X,
    const size_t incX,
    const MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > & H) [inline]
```

**11.13.1.16 ScalAndReduce() [2/3]**

```
template<class Field , class AlgoT , class ParSeqTrait >
void ScalAndReduce (
    const Field & F,
    const size_t M,
    const size_t N,
    const typename Field::Element alpha,
    typename Field::Element_ptr A,
    const size_t lda,
    const MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > & H) [inline]
```

**11.13.1.17 fsquareCommon()**

```
template<class Field >
Field::Element_ptr fsquareCommon (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const size_t n,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc) [inline]
```

**11.13.1.18 WinogradThreshold()** [1/4]

```
template<class Field >
int WinogradThreshold (
    const Field & F) [inline]
```

Computes the number of recursive levels to perform.

**Parameters**

<i>m</i>	the common dimension in the product AxB
----------	---

**11.13.1.19 WinogradThreshold()** [2/4]

```
template<>
int WinogradThreshold (
    const Givaro::Modular< float > & F) [inline]
```

**11.13.1.20 WinogradThreshold()** [3/4]

```
template<>
int WinogradThreshold (
    const Givaro::ModularBalanced< double > & F) [inline]
```

**11.13.1.21 WinogradThreshold()** [4/4]

```
template<>
int WinogradThreshold (
    const Givaro::ModularBalanced< float > & F) [inline]
```

**11.13.1.22 WinogradSteps()**

```
template<class Field >
int WinogradSteps (
    const Field & F,
    const size_t & m) [inline]
```

Computes the number of recursive levels to perform.

## Parameters

<i>m</i>	the common dimension in the product AxB
----------	---

## 11.13.1.23 DynamicPeeling()

```

template<class Field , class FieldMode >
void DynamicPeeling (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const size_t mr,
    const size_t nr,
    const size_t kr,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::Winograd, FieldMode > & H,
    const typename MMHelper< Field, MMHelperAlgo::Winograd, FieldMode >::Delayed←
Field::Element Cmin,
    const typename MMHelper< Field, MMHelperAlgo::Winograd, FieldMode >::Delayed←
Field::Element Cmax) [inline]

```

## 11.13.1.24 DynamicPeeling2()

```

template<class Field , class FieldMode >
void DynamicPeeling2 (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const size_t mr,
    const size_t nr,
    const size_t kr,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::Winograd, FieldMode > & H,

```

```

        const typename MMHelper< Field, MMHelperAlgo::Winograd, FieldMode >::Delayed←
Field::Element Cmin,
        const typename MMHelper< Field, MMHelperAlgo::Winograd, FieldMode >::Delayed←
Field::Element Cmax) [inline]

```

#### 11.13.1.25 WinogradCalc()

```

template<class Field , class FieldMode >
void WinogradCalc (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const FFLAS_TRANSPOSE tb,
    const size_t mr,
    const size_t nr,
    const size_t kr,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::Winograd, FieldMode > & H) [inline]

```

#### 11.13.1.26 fgemv\_convert()

```

template<typename FloatElement , class Field >
Field::Element_ptr fgemv_convert (
    const Field & F,
    const FFLAS_TRANSPOSE ta,
    const size_t M,
    const size_t N,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr X,
    const size_t incX,
    const typename Field::Element beta,
    typename Field::Element_ptr Y,
    const size_t incY) [inline]

```

#### 11.13.1.27 fger\_convert()

```

template<class FloatElement , class Field >
void fger_convert (
    const Field & F,
    const size_t M,
    const size_t N,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr x,
    const size_t incx,
    typename Field::ConstElement_ptr y,
    const size_t incy,
    typename Field::Element_ptr A,
    const size_t lda) [inline]

```



**11.13.1.28 fsyrk\_convert()**

```
template<class NewField , class Field , class FieldMode >
Field::Element_ptr fsyrk_convert (
    const Field & F,
    const FFLAS_UPLO UpLo,
    const FFLAS_TRANSPOSE trans,
    const size_t N,
    const size_t K,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc,
    MMHelper< Field, MMHelperAlgo::Classic, FieldMode > & H) [inline]
```

**11.13.1.29 ScalAndReduce() [3/3]**

```
template<class Field , class AlgoT , class ParSeqTrait >
void ScalAndReduce (
    const Field & F,
    const FFLAS_UPLO UpLo,
    const size_t N,
    const typename Field::Element alpha,
    typename Field::Element_ptr A,
    const size_t lda,
    const MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > & H) [inline]
```

**11.13.1.30 NeedPreScalReduction() [1/2]**

```
template<class Field , class Element , class AlgoT , class ParSeqTrait >
bool NeedPreScalReduction (
    Element & Outmin,
    Element & Outmax,
    Element & Op1min,
    Element & Op1max,
    const Element & x,
    MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > & WH) [inline]
```

**11.13.1.31 NeedPreScalReduction() [2/2]**

```
template<class Field , class Element , class AlgoT , class ModeT , class ParSeqTrait >
bool NeedPreScalReduction (
    Element & Outmin,
    Element & Outmax,
    Element & Op1min,
    Element & Op1max,
    const Element & x,
    MMHelper< Field, AlgoT, ModeT, ParSeqTrait > & WH) [inline]
```

**11.13.1.32 NeedPreAxyReduction() [1/2]**

```
template<class Field , class Element , class AlgoT , class ParSeqTrait >
bool NeedPreAxyReduction (
    Element & Outmin,
    Element & Outmax,
    Element & Op1min,
    Element & Op1max,
    Element & Op2min,
    Element & Op2max,
    const Element & x,
    MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > & WH) [inline]
```

**11.13.1.33 NeedPreAxyReduction() [2/2]**

```
template<class Field , class Element , class AlgoT , class ModeT , class ParSeqTrait >
bool NeedPreAxyReduction (
    Element & Outmin,
    Element & Outmax,
    Element & Op1min,
    Element & Op1max,
    Element & Op2min,
    Element & Op2max,
    const Element & x,
    MMHelper< Field, AlgoT, ModeT, ParSeqTrait > & WH) [inline]
```

**11.13.1.34 min\_types() [1/7]**

```
template<class DFE >
size_t min_types (
    const DFE & k) [inline]
```

**11.13.1.35 min\_types() [2/7]**

```
template<>
size_t min_types (
    const RecInt::rint< 6 > & k) [inline]
```

**11.13.1.36 min\_types() [3/7]**

```
template<>
size_t min_types (
    const RecInt::rint< 7 > & k) [inline]
```

**11.13.1.37 min\_types() [4/7]**

```
template<>
size_t min_types (
    const RecInt::rint< 8 > & k) [inline]
```

**11.13.138 min\_types()** [5/7]

```
template<>
size_t min_types (
    const RecInt::rint< 9 > & k) [inline]
```

**11.13.139 min\_types()** [6/7]

```
template<>
size_t min_types (
    const RecInt::rint< 10 > & k) [inline]
```

**11.13.140 min\_types()** [7/7]

```
template<>
size_t min_types (
    const Givaro::Integer & k) [inline]
```

**11.13.141 unfit()** [1/4]

```
template<class T >
bool unfit (
    T x) [inline]
```

**11.13.142 unfit()** [2/4]

```
template<>
bool unfit (
    int64_t x) [inline]
```

**11.13.143 unfit()** [3/4]

```
template<size_t K>
bool unfit (
    RecInt::rint< K > x) [inline]
```

**11.13.144 unfit()** [4/4]

```
template<>
bool unfit (
    RecInt::rint< 6 > x) [inline]
```

**11.13.1.45 igemm\_colmajor()** [1/2]

```
template<enum FFLAS_TRANSPOSE tA, enum FFLAS_TRANSPOSE tB>
void igemm_colmajor (
    size_t rows,
    size_t cols,
    size_t depth,
    const int64_t alpha,
    const int64_t * A,
    size_t lda,
    const int64_t * B,
    size_t ldb,
    int64_t * C,
    size_t ldc)
```

**11.13.1.46 igemm\_colmajor()** [2/2]

```
template<enum FFLAS_TRANSPOSE tA, enum FFLAS_TRANSPOSE tB, enum number_kind alpha_kind>
void igemm_colmajor (
    size_t rows,
    size_t cols,
    size_t depth,
    const int64_t alpha,
    const int64_t * A,
    size_t lda,
    const int64_t * B,
    size_t ldb,
    int64_t * C,
    size_t ldc)
```

**11.13.1.47 igemm()**

```
void igemm (
    const enum FFLAS_TRANSPOSE TransA,
    const enum FFLAS_TRANSPOSE TransB,
    size_t rows,
    size_t cols,
    size_t depth,
    const int64_t alpha,
    const int64_t * A,
    size_t lda,
    const int64_t * B,
    size_t ldb,
    const int64_t beta,
    int64_t * C,
    size_t ldc) [inline]
```

**Todo** use primitive (no `Field()`) and specialise for int64.

**Todo** use primitive (no `Field()`) and specialise for int64.

**11.13.148 MatF2MatD\_Triangular()**

```
template<class Field >
void MatF2MatD_Triangular (
    const Field & F,
    Givaro::DoubleDomain::Element_ptr S,
    const size_t lds,
    typename Field::ConstElement_ptr const E,
    const size_t lde,
    const size_t m,
    const size_t n)
```

**11.13.149 MatF2MatFI\_Triangular()**

```
template<class Field >
void MatF2MatFI_Triangular (
    const Field & F,
    Givaro::FloatDomain::Element_ptr S,
    const size_t lds,
    typename Field::ConstElement_ptr const E,
    const size_t lde,
    const size_t m,
    const size_t n)
```

**Todo** do finit(...,FFLAS\_TRANS,FFLAS\_DIAG)  
do fconvert(...,FFLAS\_TRANS,FFLAS\_DIAG)

**11.14 FFLAS::sell\_details Namespace Reference****Data Structures**

- struct [Coo](#)
- struct [Info](#)

**11.15 FFLAS::sparse\_details Namespace Reference****Functions**

- template<class Field >  
void [init\\_y](#) (const Field &F, const size\_t m, const typename Field::Element b, typename Field::Element\_ptr y)
- template<class Field >  
void [init\\_y](#) (const Field &F, const size\_t m, const size\_t n, const typename Field::Element b, typename Field::Element\_ptr y, const int ldy)
- template<class Field , class SM , class FC , class MZO >  
std::enable\_if<!std::is\_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value||std::is\_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value)>::type [fspmv\\_dispatch](#) (const Field &F, const SM &A, typename Field::ConstElement\_ptr x, typename Field::Element\_ptr y, FC fc, MZO mzo)

- `template<class Field, class SM, class FC, class MZO >`  
`std::enable_if< std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value || std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value >::type fspmv_dispatch (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FC fc, MZO mzo)`
- `template<class Field, class SM >`  
`void fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag, NotZOSparseMatrix)`
- `template<class Field, class SM >`  
`std::enable_if< !isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
- `template<class Field, class SM >`  
`std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
- `template<class Field, class SM >`  
`std::enable_if< !isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, NotZOSparseMatrix)`
- `template<class Field, class SM >`  
`std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, NotZOSparseMatrix)`
- `template<class Field, class SM >`  
`void fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag, ZOSparseMatrix)`
- `template<class Field, class SM >`  
`std::enable_if< !isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, ZOSparseMatrix)`
- `template<class Field, class SM >`  
`std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, ZOSparseMatrix)`
- `template<class Field, class SM >`  
`void fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, std::true_type)`
- `template<class Field, class SM, class FCat, class MZO >`  
`std::enable_if< !(std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value || std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value) >::type fspmm_dispatch (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FCat, MZO)`
- `template<class Field, class SM, class FCat, class MZO >`  
`std::enable_if< std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value || std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value >::type fspmm_dispatch (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FCat, MZO)`
- `template<class Field, class SM >`  
`void fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag, NotZOSparseMatrix)`
- `template<class Field, class SM >`  
`std::enable_if< support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
- `template<class Field, class SM >`  
`std::enable_if< !support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM`

- &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x, int ldx, typename [Field::Element\\_ptr](#) y, int ldy, [FieldCategories::UnparametricTag](#), [NotZOSparseMatrix](#))
- `template<class Field , class SM >`  
`std::enable_if<support\_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::ModularTag, NotZOSparseMatrix)`
  - `template<class Field , class SM >`  
`std::enable_if<!support\_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::ModularTag, NotZOSparseMatrix)`
  - `template<class Field , class SM >`  
`void fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::GenericTag, ZOSparseMatrix)`
  - `template<class Field , class SM >`  
`std::enable_if<support\_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::UnparametricTag, ZOSparseMatrix)`
  - `template<class Field , class SM >`  
`std::enable_if<!support\_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::UnparametricTag, ZOSparseMatrix)`
  - `template<class Field , class SM >`  
`void fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::ModularTag, ZOSparseMatrix)`
  - `template<class Field , class SM , class FCat , class MZO >`  
`std::enable_if<!(std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value)>::type pfspmm\_dispatch (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FCat, MZO)`
  - `template<class Field , class SM , class FCat , class MZO >`  
`std::enable_if< std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value >::type pfspmm\_dispatch (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FCat, MZO)`
  - `template<class Field , class SM >`  
`void pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::GenericTag, NotZOSparseMatrix)`
  - `template<class Field , class SM >`  
`std::enable_if<support\_simd< typenameField::Element >::value >::type pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
  - `template<class Field , class SM >`  
`std::enable_if<!support\_simd< typenameField::Element >::value >::type pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
  - `template<class Field , class SM >`  
`std::enable_if<support\_simd< typenameField::Element >::value >::type pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::ModularTag, NotZOSparseMatrix)`
  - `template<class Field , class SM >`  
`std::enable_if<!support\_simd< typenameField::Element >::value >::type pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::ModularTag, NotZOSparseMatrix)`
  - `template<class Field , class SM >`  
`void pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::GenericTag, ZOSparseMatrix)`

- `template<class Field, class SM >`  
`std::enable_if< support_simd< typenameField::Element >::value >::type pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag, ZOSparseMatrix)`
- `template<class Field, class SM >`  
`std::enable_if< !support_simd< typenameField::Element >::value >::type pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag, ZOSparseMatrix)`
- `template<class Field, class SM >`  
`void pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::ModularTag, ZOSparseMatrix)`
- `template<class Field, class SM >`  
`void pfspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag, std::false_type)`
- `template<class Field, class SM >`  
`void pfspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, std::false_type)`
- `template<class Field, class SM >`  
`void pfspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, std::false_type)`
- `template<class Field, class SM >`  
`void pfspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag, std::true_type)`
- `template<class Field, class SM >`  
`void pfspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, std::true_type)`
- `template<class Field, class SM >`  
`void pfspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, std::true_type)`
- `template<class Field, class SM >`  
`std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value &&support_simd< typenameField::Element >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
- `template<class Field, class SM >`  
`std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value &&support_simd< typenameField::Element >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, NotZOSparseMatrix)`
- `template<class Field, class SM >`  
`std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value &&support_simd< typenameField::Element >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, ZOSparseMatrix)`

## 11.15.1 Function Documentation

### 11.15.1.1 `init_y()` [1/2]

```
template<class Field >
void init_y (
    const Field & F,
    const size_t m,
    const typename Field::Element b,
    typename Field::Element_ptr y) [inline]
```



**11.15.1.2 init\_y()** [2/2]

```
template<class Field >
void init_y (
    const Field & F,
    const size_t m,
    const size_t n,
    const typename Field::Element b,
    typename Field::Element_ptr y,
    const int ldy) [inline]
```

**11.15.1.3 fspmv\_dispatch()** [1/2]

```
template<class Field , class SM , class FC , class MZO >
std::enable_if<!(std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::Mac
>::value||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineInt
>::value)>::type fspmv_dispatch (
    const Field & F,
    const SM & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FC fc,
    MZO mzo) [inline]
```

**11.15.1.4 fspmv\_dispatch()** [2/2]

```
template<class Field , class SM , class FC , class MZO >
std::enable_if< std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::Mac
>::value||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineInt
>::value >::type fspmv_dispatch (
    const Field & F,
    const SM & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FC fc,
    MZO mzo) [inline]
```

**11.15.1.5 fspmv()** [1/12]

```
template<class Field , class SM >
void fspmv (
    const Field & F,
    const SM & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::GenericTag ,
    NotZOSparseMatrix ) [inline]
```

**11.15.1.6 fspmv()** [2/12]

```
template<class Field , class SM >
std::enable_if<!isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (
    const Field & F,
    const SM & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::UnparametricTag ,
    NotZOSparseMatrix ) [inline]
```

**11.15.1.7 fspmv()** [3/12]

```
template<class Field , class SM >
std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (
    const Field & F,
    const SM & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::UnparametricTag ,
    NotZOSparseMatrix ) [inline]
```

**11.15.1.8 fspmv()** [4/12]

```
template<class Field , class SM >
std::enable_if<!isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (
    const Field & F,
    const SM & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::ModularTag ,
    NotZOSparseMatrix ) [inline]
```

**11.15.1.9 fspmv()** [5/12]

```
template<class Field , class SM >
std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (
    const Field & F,
    const SM & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::ModularTag ,
    NotZOSparseMatrix ) [inline]
```

**11.15.1.10 fspmv()** [6/12]

```
template<class Field , class SM >
void fspmv (
    const Field & F,
    const SM & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::GenericTag ,
    ZOSparseMatrix ) [inline]
```

**11.15.1.11 fspmv()** [7/12]

```
template<class Field , class SM >
std::enable_if<!isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (
    const Field & F,
    const SM & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::UnparametricTag ,
    ZOSparseMatrix ) [inline]
```

**11.15.1.12 fspmv()** [8/12]

```
template<class Field , class SM >
std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (
    const Field & F,
    const SM & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::UnparametricTag ,
    ZOSparseMatrix ) [inline]
```

**11.15.1.13 fspmv()** [9/12]

```
template<class Field , class SM >
void fspmv (
    const Field & F,
    const SM & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::ModularTag ,
    std::true_type ) [inline]
```

**11.15.1.14 fspmm\_dispatch()** [1/2]

```
template<class Field , class SM , class FCat , class MZO >
std::enable_if<!(std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::Ma
>::value||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineInt
>::value)>::type fspmm_dispatch (
    const Field & F,
    const SM & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FCat ,
    MZO ) [inline]
```

**11.15.1.15 fspmm\_dispatch() [2/2]**

```

template<class Field , class SM , class FCat , class MZO >
std::enable_if< std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::Mac
>::value||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineInt
>::value >::type fspmm_dispatch (
    const Field & F,
    const SM & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FCat ,
    MZO ) [inline]

```

**11.15.1.16 fspmm() [1/9]**

```

template<class Field , class SM >
void fspmm (
    const Field & F,
    const SM & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::GenericTag ,
    NotZOSparseMatrix ) [inline]

```

**11.15.1.17 fspmm() [2/9]**

```

template<class Field , class SM >
std::enable_if< support_simd< typenameField::Element >::value >::type fspmm (
    const Field & F,
    const SM & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::UnparametricTag ,
    NotZOSparseMatrix ) [inline]

```

**11.15.1.18 fspmm() [3/9]**

```

template<class Field , class SM >
std::enable_if< !support_simd< typenameField::Element >::value >::type fspmm (
    const Field & F,
    const SM & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::UnparametricTag ,
    NotZOSparseMatrix ) [inline]

```

**11.15.1.19 fspmm() [4/9]**

```
template<class Field , class SM >
std::enable_if< support_simd< typenameField::Element >::value >::type fspmm (
    const Field & F,
    const SM & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::ModularTag ,
    NotZOSparseMatrix ) [inline]
```

**11.15.1.20 fspmm() [5/9]**

```
template<class Field , class SM >
std::enable_if< !support_simd< typenameField::Element >::value >::type fspmm (
    const Field & F,
    const SM & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::ModularTag ,
    NotZOSparseMatrix ) [inline]
```

**11.15.1.21 fspmm() [6/9]**

```
template<class Field , class SM >
void fspmm (
    const Field & F,
    const SM & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::GenericTag ,
    ZOSparseMatrix ) [inline]
```

**11.15.1.22 fspmm() [7/9]**

```
template<class Field , class SM >
std::enable_if< support_simd< typenameField::Element >::value >::type fspmm (
    const Field & F,
    const SM & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::UnparametricTag ,
    ZOSparseMatrix ) [inline]
```

**11.15.1.23 fspmm()** [8/9]

```
template<class Field , class SM >
std::enable_if<!support\_simd< typenameField::Element >::value >::type fspmm (
    const Field & F,
    const SM & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::UnparametricTag ,
    ZOSparseMatrix ) [inline]
```

**11.15.1.24 fspmm()** [9/9]

```
template<class Field , class SM >
void fspmm (
    const Field & F,
    const SM & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::ModularTag ,
    ZOSparseMatrix ) [inline]
```

**11.15.1.25 pfspmm\_dispatch()** [1/2]

```
template<class Field , class SM , class FCat , class MZO >
std::enable_if<!(std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::Ma
>::value||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineInt
>::value)>::type pfspmm_dispatch (
    const Field & F,
    const SM & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FCat ,
    MZO ) [inline]
```

**11.15.1.26 pfspmm\_dispatch()** [2/2]

```
template<class Field , class SM , class FCat , class MZO >
std::enable_if< std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::Mac
>::value||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineInt
>::value >::type pfspmm_dispatch (
    const Field & F,
    const SM & A,
```

```

    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FCat ,
    MZO ) [inline]

```

#### 11.15.1.27 pfspmm() [1/9]

```

template<class Field , class SM >
void pfspmm (
    const Field & F,
    const SM & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::GenericTag ,
    NotZOSparseMatrix ) [inline]

```

#### 11.15.1.28 pfspmm() [2/9]

```

template<class Field , class SM >
std::enable_if< support_simd< typenameField::Element >::value >::type pfspmm (
    const Field & F,
    const SM & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::UnparametricTag ,
    NotZOSparseMatrix ) [inline]

```

#### 11.15.1.29 pfspmm() [3/9]

```

template<class Field , class SM >
std::enable_if< !support_simd< typenameField::Element >::value >::type pfspmm (
    const Field & F,
    const SM & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::UnparametricTag ,
    NotZOSparseMatrix ) [inline]

```

**11.15.1.30 pfspmm()** [4/9]

```
template<class Field , class SM >
std::enable_if< support_simd< typenameField::Element >::value >::type pfspmm (
    const Field & F,
    const SM & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::ModularTag ,
    NotZOSparseMatrix ) [inline]
```

**11.15.1.31 pfspmm()** [5/9]

```
template<class Field , class SM >
std::enable_if< !support_simd< typenameField::Element >::value >::type pfspmm (
    const Field & F,
    const SM & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::ModularTag ,
    NotZOSparseMatrix ) [inline]
```

**11.15.1.32 pfspmm()** [6/9]

```
template<class Field , class SM >
void pfspmm (
    const Field & F,
    const SM & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::GenericTag ,
    ZOSparseMatrix ) [inline]
```

**11.15.1.33 pfspmm()** [7/9]

```
template<class Field , class SM >
std::enable_if< support_simd< typenameField::Element >::value >::type pfspmm (
    const Field & F,
    const SM & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::UnparametricTag ,
    ZOSparseMatrix ) [inline]
```



**11.15.134 pfspmm()** [8/9]

```
template<class Field , class SM >
std::enable_if<!support\_simd< typenameField::Element >::value >::type pfspmm (
    const Field & F,
    const SM & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::UnparametricTag ,
    ZOSparseMatrix ) [inline]
```

**11.15.135 pfspmm()** [9/9]

```
template<class Field , class SM >
void pfspmm (
    const Field & F,
    const SM & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::ModularTag ,
    ZOSparseMatrix ) [inline]
```

**11.15.136 pfspmv()** [1/6]

```
template<class Field , class SM >
void pfspmv (
    const Field & F,
    const SM & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::GenericTag ,
    std::false_type ) [inline]
```

**11.15.137 pfspmv()** [2/6]

```
template<class Field , class SM >
void pfspmv (
    const Field & F,
    const SM & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::UnparametricTag ,
    std::false_type ) [inline]
```

**11.15.1.38 pfspmv()** [3/6]

```
template<class Field , class SM >
void pfspmv (
    const Field & F,
    const SM & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::ModularTag ,
    std::false_type ) [inline]
```

**11.15.1.39 pfspmv()** [4/6]

```
template<class Field , class SM >
void pfspmv (
    const Field & F,
    const SM & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::GenericTag ,
    std::true_type ) [inline]
```

**11.15.1.40 pfspmv()** [5/6]

```
template<class Field , class SM >
void pfspmv (
    const Field & F,
    const SM & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::UnparametricTag ,
    std::true_type ) [inline]
```

**11.15.1.41 pfspmv()** [6/6]

```
template<class Field , class SM >
void pfspmv (
    const Field & F,
    const SM & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::ModularTag ,
    std::true_type ) [inline]
```

**11.15.1.42 fspmv()** [10/12]

```
template<class Field , class SM >
std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value &&support_simd< typenameField←
::Element >::value >::type fspmv (
    const Field & F,
    const SM & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::UnparametricTag ,
    NotZOSparseMatrix ) [inline]
```

**11.15.1.43 fspmv()** [11/12]

```
template<class Field , class SM >
std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value &&support_simd< typenameField↵
::Element >::value >::type fspmv (
    const Field & F,
    const SM & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::ModularTag ,
    NotZOSparseMatrix ) [inline]
```

**11.15.1.44 fspmv()** [12/12]

```
template<class Field , class SM >
std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value &&support_simd< typenameField↵
::Element >::value >::type fspmv (
    const Field & F,
    const SM & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::UnparametricTag ,
    ZOSparseMatrix ) [inline]
```

**11.16 FFLAS::sparse\_details\_impl Namespace Reference****Functions**

- template<class Field >  
void fspmm (const Field &F, const Sparse< Field, SparseMatrix\_t::COO > &A, size\_t blockSize, typename Field::ConstElement\_ptr x\_, int ldx, typename Field::Element\_ptr y\_, int ldy, FieldCategories::GenericTag)
- template<class Field >  
void fspmm (const Field &F, const Sparse< Field, SparseMatrix\_t::COO > &A, size\_t blockSize, typename Field::ConstElement\_ptr x\_, int ldx, typename Field::Element\_ptr y\_, int ldy, FieldCategories::UnparametricTag)
- template<class Field >  
void fspmm (const Field &F, const Sparse< Field, SparseMatrix\_t::COO > &A, size\_t blockSize, typename Field::ConstElement\_ptr x\_, int ldx, typename Field::Element\_ptr y\_, int ldy, const int64\_t kmax)
- template<class Field >  
void fspmm\_simd\_aligned (const Field &F, const Sparse< Field, SparseMatrix\_t::COO > &A, size\_t block↵  
Size, typename Field::ConstElement\_ptr x\_, int ldx, typename Field::Element\_ptr y\_, int ldy, const int64\_t  
kmax)
- template<class Field >  
void fspmm\_simd\_unaligned (const Field &F, const Sparse< Field, SparseMatrix\_t::COO > &A, size\_↵  
t blockSize, typename Field::ConstElement\_ptr x\_, int ldx, typename Field::Element\_ptr y\_, int ldy, const  
int64\_t kmax)
- template<class Field >  
void fspmm\_one (const Field &F, const Sparse< Field, SparseMatrix\_t::COO\_ZO > &A, size\_↵  
t blockSize, typename Field::ConstElement\_ptr x\_, int ldx, typename Field::Element\_ptr y\_, int ldy,  
FieldCategories::GenericTag)
- template<class Field >  
void fspmm\_mone (const Field &F, const Sparse< Field, SparseMatrix\_t::COO\_ZO > &A, size\_↵  
t blockSize, typename Field::ConstElement\_ptr x\_, int ldx, typename Field::Element\_ptr y\_, int ldy,  
FieldCategories::GenericTag)

- `template<class Field >`  
`void fspmm_one_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_one_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_mone_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_mone_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::COO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::COO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::COO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const uint64_t kmax)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, const int64_t kmax)`
- `template<class Field >`  
`void pfspmm_one (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmm_mone (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmm_one (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`

- template<class [Field](#) >  
void [pfspmm\\_mone](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [pfspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [pfspmv\\_task](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, const [index\\_t](#) iStart, const [index\\_t](#) iStop, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [pfspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [pfspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, const int64\_t kmax)
- template<class [Field](#) >  
void [pfspmv\\_one](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [pfspmv\\_mone](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [pfspmv\\_one](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [pfspmv\\_mone](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [fspmm](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [fspmm](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A, [index\\_t](#) blockSize, typename [Field::ConstElement\\_ptr](#) x\_, [index\\_t](#) ldx, typename [Field::Element\\_ptr](#) y\_, [index\\_t](#) ldy, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [fspmm\\_simd\\_aligned](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [fspmm\\_simd\\_unaligned](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [fspmm](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, const int64\_t kmax)
- template<class [Field](#) >  
void [fspmm\\_one](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [fspmm\\_mone](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [fspmm\\_one\\_simd\\_aligned](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A,

- ```
size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy,
FieldCategories::UnparametricTag)
```
- `template<class Field >`  
`void fspmm_one_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A,`  
`size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy,`  
`FieldCategories::UnparametricTag)`
  - `template<class Field >`  
`void fspmm_mone_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A,`  
`size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy,`  
`FieldCategories::UnparametricTag)`
  - `template<class Field >`  
`void fspmm_mone_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A,`  
`size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy,`  
`FieldCategories::UnparametricTag)`
  - `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, typename Field::ConstElement_ptr`  
`x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
  - `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, typename Field::ConstElement_ptr`  
`x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
  - `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, typename Field::ConstElement_ptr`  
`x_, typename Field::Element_ptr y_, const int64_t kmax)`
  - `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
  - `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
  - `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
  - `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
  - `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, type-`  
`name Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag)`
  - `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, type-`  
`name Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag)`
  - `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, type-`  
`name Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag)`
  - `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_↵`  
`t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy,`  
`FieldCategories::UnparametricTag)`
  - `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, type-`  
`name Field::ConstElement_ptr x, typename Field::Element_ptr y, const int64_t kmax)`
  - `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, type-`  
`name Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, const int64_t kmax)`
  - `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`

- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const int64_t kmax)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, const int64_t kmax)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const uint64_t kmax)`
- `template<class Field >`  
`void pfspm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, const int64_t kmax)`
- `template<class Field >`  
`void pfspm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, const int64_t kmax)`
- `template<class Field, class Func >`  
`void pfspm_zo (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, Func &&func)`
- `template<class Field, class Func >`  
`void pfspm_zo (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, Func &&func)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const int64_t kmax)`



- `template<class Field >`  
`void pfspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, const int64_t kmax)`
- `template<class Field >`  
`void fspmm_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_one_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_one_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_mone_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_mone_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`



- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const uint64_t kmax)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const uint64_t kmax)`
- `template<class Field >`  
`void pfspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_simd (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_simd (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const uint64_t kmax)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const uint64_t kmax)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`

- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_one_simd (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_mone_simd (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, uint64_t kmax)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, uint64_t kmax)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, uint64_t kmax)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, uint64_t kmax)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::SELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::SELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`

- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::SELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const int64_t kmax)`
- `template<class Field >`  
`void pfspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::SELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_simd (const Field &F, const Sparse< Field, SparseMatrix_t::SELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::SELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_simd (const Field &F, const Sparse< Field, SparseMatrix_t::SELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const uint64_t kmax)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::SELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const uint64_t kmax)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_one_simd (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_mone_simd (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`

## 11.16.1 Function Documentation

### 11.16.1.1 fspmm() [1/15]

```
template<class Field >
void fspmm (
```

```

    const Field & F,
    const Sparse< Field, SparseMatrix_t::COO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::GenericTag ) [inline]

```

#### 11.16.1.2 fspmm() [2/15]

```

template<class Field >
void fspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::COO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]

```

#### 11.16.1.3 fspmm() [3/15]

```

template<class Field >
void fspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::COO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    const int64_t kmax) [inline]

```

#### 11.16.1.4 fspmm\_simd\_aligned() [1/2]

```

template<class Field >
void fspmm_simd_aligned (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::COO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    const int64_t kmax) [inline]

```

**11.16.1.5 fspmm\_simd\_unaligned()** [1/2]

```
template<class Field >
void fspmm_simd_unaligned (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::COO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    const int64_t kmax) [inline]
```

**11.16.1.6 fspmm\_one()** [1/4]

```
template<class Field >
void fspmm_one (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::COO_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.7 fspmm\_mone()** [1/4]

```
template<class Field >
void fspmm_mone (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::COO_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.8 fspmm\_one\_simd\_aligned()** [1/3]

```
template<class Field >
void fspmm_one_simd_aligned (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::COO_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.9 fspmm\_one\_simd\_unaligned()** [1/3]

```
template<class Field >
void fspmm_one_simd_unaligned (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::COO_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.10 fspmm\_mone\_simd\_aligned()** [1/3]

```
template<class Field >
void fspmm_mone_simd_aligned (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::COO_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.11 fspmm\_mone\_simd\_unaligned()** [1/3]

```
template<class Field >
void fspmm_mone_simd_unaligned (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::COO_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.12 fspmv()** [1/21]

```
template<class Field >
void fspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::COO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.13 fspmv()** [2/21]

```
template<class Field >
void fspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::COO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.14 fspmv()** [3/21]

```
template<class Field >
void fspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::COO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    const uint64_t kmax) [inline]
```

**11.16.1.15 fspmv\_one()** [1/10]

```
template<class Field >
void fspmv_one (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::COO_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.16 fspmv\_mone()** [1/10]

```
template<class Field >
void fspmv_mone (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::COO_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.17 fspmv\_one()** [2/10]

```
template<class Field >
void fspmv_one (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::COO_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.18 fspmv\_mone()** [2/10]

```
template<class Field >
void fspmv_mone (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::COO_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.19 pfspmm()** [1/18]

```
template<class Field >
void pfspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.20 pfspmm()** [2/18]

```
template<class Field >
void pfspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.21 pfspmm()** [3/18]

```
template<class Field >
void pfspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    const int64_t kmax) [inline]
```



**11.16.1.22 pfspmm\_one() [1/2]**

```
template<class Field >
void pfspmm_one (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.23 pfspmm\_mone() [1/2]**

```
template<class Field >
void pfspmm_mone (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.24 pfspmm\_one() [2/2]**

```
template<class Field >
void pfspmm_one (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.25 pfspmm\_mone() [2/2]**

```
template<class Field >
void pfspmm_mone (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.26 pfspmv()** [1/18]

```
template<class Field >
void pfspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.27 pfspmv\_task()**

```
template<class Field >
void pfspmv_task (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    const index_t iStart,
    const index_t iStop,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.28 pfspmv()** [2/18]

```
template<class Field >
void pfspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.29 pfspmv()** [3/18]

```
template<class Field >
void pfspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    const int64_t kmax) [inline]
```

**11.16.1.30 pfspmv\_one()** [1/8]

```
template<class Field >
void pfspmv_one (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.31 pfspmv\_mone()** [1/8]

```
template<class Field >
void pfspmv_mone (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.32 pfspmv\_one()** [2/8]

```
template<class Field >
void pfspmv_one (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.33 pfspmv\_mone()** [2/8]

```
template<class Field >
void pfspmv_mone (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.34 fspmm()** [4/15]

```
template<class Field >
void fspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.35 fspmm()** [5/15]

```
template<class Field >
void fspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR > & A,
    index_t blockSize,
    typename Field::ConstElement_ptr x_,
    index_t ldx,
    typename Field::Element_ptr y_,
    index_t ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.36 fspmm\_simd\_aligned()** [2/2]

```
template<class Field >
void fspmm_simd_aligned (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.37 fspmm\_simd\_unaligned()** [2/2]

```
template<class Field >
void fspmm_simd_unaligned (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.38 fspmm()** [6/15]

```
template<class Field >
void fspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    const int64_t kmax) [inline]
```

**11.16.1.39 fspmm\_one()** [2/4]

```
template<class Field >
void fspmm_one (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::GenericTag ) [inline]
```

**11.16.140 fspmm\_mone()** [2/4]

```
template<class Field >
void fspmm_mone (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::GenericTag ) [inline]
```

**11.16.141 fspmm\_one\_simd\_aligned()** [2/3]

```
template<class Field >
void fspmm_one_simd_aligned (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.142 fspmm\_one\_simd\_unaligned()** [2/3]

```
template<class Field >
void fspmm_one_simd_unaligned (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.143 fspmm\_mone\_simd\_aligned()** [2/3]

```
template<class Field >
void fspmm_mone_simd_aligned (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.44 fspmm\_mone\_simd\_unaligned()** [2/3]

```
template<class Field >
void fspmm_mone_simd_unaligned (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldz,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.45 fspmv()** [4/21]

```
template<class Field >
void fspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.46 fspmv()** [5/21]

```
template<class Field >
void fspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.47 fspmv()** [6/21]

```
template<class Field >
void fspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    const int64_t kmax) [inline]
```

**11.16.1.48 fspmv\_one()** [3/10]

```
template<class Field >
void fspmv_one (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.49 fspmv\_mone()** [3/10]

```
template<class Field >
void fspmv_mone (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.50 fspmv\_one()** [4/10]

```
template<class Field >
void fspmv_one (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.51 fspmv\_mone()** [4/10]

```
template<class Field >
void fspmv_mone (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.52 pfspmm()** [4/18]

```
template<class Field >
void pfspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.53 pfspmm()** [5/18]

```
template<class Field >
void pfspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::GenericTag ) [inline]
```

**11.16.154 pfspmm()** [6/18]

```
template<class Field >
void pfspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.155 pfspmm()** [7/18]

```
template<class Field >
void pfspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.156 pfspmm()** [8/18]

```
template<class Field >
void pfspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    const int64_t kmax) [inline]
```

**11.16.157 pfspmm()** [9/18]

```
template<class Field >
void pfspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    const int64_t kmax) [inline]
```



**11.16.1.58 pfspmv()** [4/18]

```
template<class Field >
void pfspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.59 pfspmv()** [5/18]

```
template<class Field >
void pfspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.60 pfspmv()** [6/18]

```
template<class Field >
void pfspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    const int64_t kmax) [inline]
```

**11.16.1.61 fspmm()** [7/15]

```
template<class Field >
void fspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.62 fspmm()** [8/15]

```
template<class Field >
void fspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.63 fspmm()** [9/15]

```
template<class Field >
void fspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    const int64_t kmax) [inline]
```

**11.16.1.64 fspmv()** [7/21]

```
template<class Field >
void fspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.65 fspmv()** [8/21]

```
template<class Field >
void fspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.66 fspmv()** [9/21]

```
template<class Field >
void fspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    const uint64_t kmax) [inline]
```

**11.16.1.67 pfspmm()** [10/18]

```
template<class Field >
void pfspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.68 pfspmm()** [11/18]

```
template<class Field >
void pfspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.69 pfspmm()** [12/18]

```
template<class Field >
void pfspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.70 pfspmm()** [13/18]

```
template<class Field >
void pfspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.71 pfspmm()** [14/18]

```
template<class Field >
void pfspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    const int64_t kmax) [inline]
```

**11.16.1.72 pfspmm()** [15/18]

```
template<class Field >
void pfspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    const int64_t kmax) [inline]
```

**11.16.1.73 pfspmm\_zo()** [1/2]

```
template<class Field , class Func >
void pfspmm_zo (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    Func && func) [inline]
```

**11.16.1.74 pfspmm\_zo()** [2/2]

```
template<class Field , class Func >
void pfspmm_zo (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    Func && func) [inline]
```

**11.16.1.75 pfspmv()** [7/18]

```
template<class Field >
void pfspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.76 pfspmv()** [8/18]

```
template<class Field >
void pfspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.77 pfspmv()** [9/18]

```
template<class Field >
void pfspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    const int64_t kmax) [inline]
```

**11.16.1.78 pfspmv\_one()** [3/8]

```
template<class Field >
void pfspmv_one (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.79 pfspmv\_mone()** [3/8]

```
template<class Field >
void pfspmv_mone (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.80 pfspmv\_one()** [4/8]

```
template<class Field >
void pfspmv_one (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.81 pfspmv\_mone()** [4/8]

```
template<class Field >
void pfspmv_mone (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.82 fspmm()** [10/15]

```
template<class Field >
void fspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.83 fspmm()** [11/15]

```
template<class Field >
void fspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.84 fspmm()** [12/15]

```
template<class Field >
void fspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    const int64_t kmax) [inline]
```

**11.16.1.85 fspmm\_mone()** [3/4]

```
template<class Field >
void fspmm_mone (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.86 fspmm\_one()** [3/4]

```
template<class Field >
void fspmm_one (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.87 fspmm\_mone()** [4/4]

```
template<class Field >
void fspmm_mone (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.88 fspmm\_one()** [4/4]

```
template<class Field >
void fspmm_one (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.89 fspmm\_one\_simd\_aligned()** [3/3]

```
template<class Field >
void fspmm_one_simd_aligned (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.90 fspmm\_one\_simd\_unaligned()** [3/3]

```
template<class Field >
void fspmm_one_simd_unaligned (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.91 fspmm\_mone\_simd\_aligned()** [3/3]

```
template<class Field >
void fspmm_mone_simd_aligned (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.92 fspmm\_mone\_simd\_unaligned()** [3/3]

```
template<class Field >
void fspmm_mone_simd_unaligned (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x_,
    int ldx,
    typename Field::Element_ptr y_,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.93 fspmv()** [10/21]

```
template<class Field >
void fspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```



**11.16.1.94 fspmv()** [11/21]

```
template<class Field >
void fspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.95 fspmv()** [12/21]

```
template<class Field >
void fspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    const uint64_t kmax) [inline]
```

**11.16.1.96 fspmv\_one()** [5/10]

```
template<class Field >
void fspmv_one (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.97 fspmv\_mone()** [5/10]

```
template<class Field >
void fspmv_mone (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.98 fspmv\_one()** [6/10]

```
template<class Field >
void fspmv_one (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.99 fspmv\_mone()** [6/10]

```
template<class Field >
void fspmv_mone (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.100 pfspmv()** [10/18]

```
template<class Field >
void pfspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_simd > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.101 pfspmv()** [11/18]

```
template<class Field >
void pfspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_simd > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.102 pfspmv()** [12/18]

```
template<class Field >
void pfspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_simd > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    const uint64_t kmax) [inline]
```

**11.16.1.103 pfspmv\_one()** [5/8]

```
template<class Field >
void pfspmv_one (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.104 pfspmv\_mone()** [5/8]

```
template<class Field >
void pfspmv_mone (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.105 pfspmv\_one()** [6/8]

```
template<class Field >
void pfspmv_one (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.106 pfspmv\_mone()** [6/8]

```
template<class Field >
void pfspmv_mone (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.107 fspmv()** [13/21]

```
template<class Field >
void fspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_simd > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.108 fspmv\_simd()** [1/4]

```
template<class Field >
void fspmv_simd (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_simd > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.109 fspmv()** [14/21]

```
template<class Field >
void fspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_simd > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.110 fspmv\_simd()** [2/4]

```
template<class Field >
void fspmv_simd (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_simd > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    const uint64_t kmax) [inline]
```

**11.16.1.111 fspmv()** [15/21]

```
template<class Field >
void fspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_simd > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    const uint64_t kmax) [inline]
```

**11.16.1.112 fspmv\_one()** [7/10]

```
template<class Field >
void fspmv_one (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.113 fspmv\_mone()** [7/10]

```
template<class Field >
void fspmv_mone (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.114 fspmv\_one()** [8/10]

```
template<class Field >
void fspmv_one (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.115 fspmv\_mone()** [8/10]

```
template<class Field >
void fspmv_mone (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.116 fspmv\_one\_simd()** [1/2]

```
template<class Field >
void fspmv_one_simd (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.117 fspmv\_mone\_simd()** [1/2]

```
template<class Field >
void fspmv_mone_simd (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.118 pfspmm()** [16/18]

```
template<class Field >
void pfspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.119 pfsppmm()** [17/18]

```
template<class Field >
void pfsppmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.120 pfsppmm()** [18/18]

```
template<class Field >
void pfsppmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    uint64_t kmax) [inline]
```

**11.16.1.121 pfsppmv()** [13/18]

```
template<class Field >
void pfsppmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.122 pfsppmv()** [14/18]

```
template<class Field >
void pfsppmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.123 pfsppmv()** [15/18]

```
template<class Field >
void pfsppmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    uint64_t kmax) [inline]
```

**11.16.1.124 fspmm()** [13/15]

```
template<class Field >
void fspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.125 fspmm()** [14/15]

```
template<class Field >
void fspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.126 fspmm()** [15/15]

```
template<class Field >
void fspmm (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
    size_t blockSize,
    typename Field::ConstElement_ptr x,
    int ldx,
    typename Field::Element_ptr y,
    int ldy,
    uint64_t kmax) [inline]
```

**11.16.1.127 fspmv()** [16/21]

```
template<class Field >
void fspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.128 fspmv()** [17/21]

```
template<class Field >
void fspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.129 fspmv()** [18/21]

```
template<class Field >
void fspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
    typename Field::ConstElement_ptr x,
    typename Field::Element_ptr y,
    uint64_t kmax) [inline]
```

**11.16.1.130 pfspmv()** [16/18]

```
template<class Field >
void pfspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::SELL > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.131 pfspmv()** [17/18]

```
template<class Field >
void pfspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::SELL > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.132 pfspmv()** [18/18]

```
template<class Field >
void pfspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::SELL > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    const int64_t kmax) [inline]
```



**11.16.1.133 pfspmv\_one()** [7/8]

```
template<class Field >
void pfspmv_one (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::SELL_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.134 pfspmv\_mone()** [7/8]

```
template<class Field >
void pfspmv_mone (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::SELL_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.135 pfspmv\_one()** [8/8]

```
template<class Field >
void pfspmv_one (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::SELL_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.136 pfspmv\_mone()** [8/8]

```
template<class Field >
void pfspmv_mone (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::SELL_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.137 fspmv()** [19/21]

```
template<class Field >
void fspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::SELL > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.138 fspmv\_simd()** [3/4]

```
template<class Field >
void fspmv_simd (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::SELL > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.139 fspmv()** [20/21]

```
template<class Field >
void fspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::SELL > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.140 fspmv\_simd()** [4/4]

```
template<class Field >
void fspmv_simd (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::SELL > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    const uint64_t kmax) [inline]
```

**11.16.1.141 fspmv()** [21/21]

```
template<class Field >
void fspmv (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::SELL > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    const uint64_t kmax) [inline]
```

**11.16.1.142 fspmv\_one()** [9/10]

```
template<class Field >
void fspmv_one (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::SELL_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.143 fspmv\_mone()** [9/10]

```
template<class Field >
void fspmv_mone (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::SELL_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::GenericTag ) [inline]
```

**11.16.1.144 fspmv\_one\_simd()** [2/2]

```
template<class Field >
void fspmv_one_simd (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::SELL_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.145 fspmv\_mone\_simd()** [2/2]

```
template<class Field >
void fspmv_mone_simd (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::SELL_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.146 fspmv\_one()** [10/10]

```
template<class Field >
void fspmv_one (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::SELL_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

**11.16.1.147 fspmv\_mone()** [10/10]

```
template<class Field >
void fspmv_mone (
    const Field & F,
    const Sparse< Field, SparseMatrix_t::SELL_ZO > & A,
    typename Field::ConstElement_ptr x_,
    typename Field::Element_ptr y_,
    FieldCategories::UnparametricTag ) [inline]
```

## 11.17 FFLAS::StrategyParameter Namespace Reference

### Data Structures

- struct [Fixed](#)
- struct [Grain](#)
- struct [Threads](#)
- struct [ThreeD](#)
- struct [ThreeDAdaptive](#)
- struct [ThreeDInPlace](#)
- struct [TwoD](#)
- struct [TwoDAdaptive](#)

## 11.18 FFLAS::StructureHelper Namespace Reference

[StructureHelper](#) for ftrsm.

### Data Structures

- struct [Hybrid](#)
- struct [Iterative](#)
- struct [Recursive](#)

### 11.18.1 Detailed Description

[StructureHelper](#) for ftrsm.

## 11.19 FFLAS::vectorised Namespace Reference

### Namespaces

- namespace [unswitch](#)

### Data Structures

- struct [HelperMod](#)
- struct [HelperMod](#)< Field, ElementCategories::MachineIntTag >
- struct [HelperMod](#)< Field, FFLAS::ElementCategories::ArbitraryPrecIntTag >
- struct [HelperMod](#)< Field, FFLAS::ElementCategories::FixedPrecIntTag >
- struct [HelperMod](#)< Field, FFLAS::ElementCategories::MachineFloatTag >

## Functions

- `template<class SimdT , class Element , bool positive>`  
`std::enable_if< is_simd< SimdT >::value, void >::type VEC_ADD (SimdT &C, SimdT &A, SimdT &B, SimdT &Q, SimdT &T, SimdT &P, SimdT &NEGP, SimdT &MIN, SimdT &MAX)`
- `template<bool positive, class Element , class T1 , class T2 >`  
`std::enable_if< FFLAS::support_simd_add< Element >::value, void >::type addp (Element *T, const Element *TA, const Element *TB, size_t n, Element p, T1 min_, T2 max_)`
- `template<class SimdT , class Element , bool positive>`  
`std::enable_if< is_simd< SimdT >::value, void >::type VEC_SUB (SimdT &C, SimdT &A, SimdT &B, SimdT &Q, SimdT &T, SimdT &P, SimdT &NEGP, SimdT &MIN, SimdT &MAX)`
- `template<bool positive, class Element , class T1 , class T2 >`  
`std::enable_if< FFLAS::support_simd_add< Element >::value, void >::type subp (Element *T, const Element *TA, const Element *TB, const size_t n, const Element p, const T1 min_, const T2 max_)`
- `template<class Element >`  
`std::enable_if< FFLAS::support_simd_add< Element >::value, void >::type add (Element *T, const Element *TA, const Element *TB, size_t n)`
- `template<class Element >`  
`std::enable_if< FFLAS::support_simd_add< Element >::value, void >::type sub (Element *T, const Element *TA, const Element *TB, size_t n)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type axpyp (const Field &F, const typename Field::Element a, typename Field::ConstElement_ptr X, typename Field::Element_ptr Y, const size_t n)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type axpyp (const Field &F, const typename Field::Element a, typename Field::ConstElement_ptr X, typename Field::Element_ptr Y, const size_t n, const size_t incX, const size_t incY)`
- `template<class T >`  
`std::enable_if< !std::is_integral< T >::value, T >::type reduce (T A, T B)`
- `template<class T >`  
`std::enable_if< std::is_integral< T >::value, T >::type reduce (T A, T B)`
- `template<> Givaro::Integer reduce (Givaro::Integer A, Givaro::Integer B)`
- `float reduce (float A, float B, float invB, float min, float max)`
- `double reduce (double A, double B, double invB, double min, double max)`
- `int64_t reduce (int64_t A, int64_t p, double invp, double min, double max, int64_t pow50rem)`
- `template<class Field >`  
`Field::Element reduce (typename Field::Element A, HelperMod< Field, ElementCategories::MachineIntTag > &H)`
- `template<class Field >`  
`Field::Element reduce (typename Field::Element A, HelperMod< Field, ElementCategories::MachineFloatTag > &H)`
- `template<class Field >`  
`Field::Element reduce (typename Field::Element A, HelperMod< Field, ElementCategories::ArbitraryPrecIntTag > &H)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type modp (const Field &F, typename Field::ConstElement_ptr U, const size_t &n, typename Field::Element_ptr T)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type modp (const Field &F, typename Field::ConstElement_ptr U, const size_t &n, const size_t &incX, typename Field::Element_ptr T)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp (const Field &F, typename Field::Element_ptr T, const typename Field::Element alpha, typename Field::ConstElement_ptr U, const size_t n)`

- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp`  
`(const Field &F, typename Field::Element_ptr T, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr U, const size_t n, const size_t &incX)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp`  
`(const Field &F, typename Field::Element_ptr T, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr U, const size_t n, const size_t &incX, const size_t &incY)`

## 11.19.1 Function Documentation

### 11.19.1.1 VEC\_ADD()

```
template<class SimdT , class Element , bool positive>
std::enable_if< is_simd< SimdT >::value, void >::type VEC_ADD (
    SimdT & C,
    SimdT & A,
    SimdT & B,
    SimdT & Q,
    SimdT & T,
    SimdT & P,
    SimdT & NEGP,
    SimdT & MIN,
    SimdT & MAX) [inline]
```

### 11.19.1.2 addp()

```
template<bool positive, class Element , class T1 , class T2 >
std::enable_if< FFLAS::support_simd_add< Element >::value, void >::type addp (
    Element * T,
    const Element * TA,
    const Element * TB,
    size_t n,
    Element p,
    T1 min_,
    T2 max_) [inline]
```

### 11.19.1.3 VEC\_SUB()

```
template<class SimdT , class Element , bool positive>
std::enable_if< is_simd< SimdT >::value, void >::type VEC_SUB (
    SimdT & C,
    SimdT & A,
    SimdT & B,
    SimdT & Q,
    SimdT & T,
    SimdT & P,
    SimdT & NEGP,
    SimdT & MIN,
    SimdT & MAX) [inline]
```

#### 11.19.1.4 subp()

```
template<bool positive, class Element , class T1 , class T2 >
std::enable_if< FFLAS::support_simd_add< Element >::value, void >::type subp (
    Element * T,
    const Element * TA,
    const Element * TB,
    const size_t n,
    const Element p,
    const T1 min_,
    const T2 max_) [inline]
```

#### 11.19.1.5 add()

```
template<class Element >
std::enable_if< FFLAS::support_simd_add< Element >::value, void >::type add (
    Element * T,
    const Element * TA,
    const Element * TB,
    size_t n) [inline]
```

#### 11.19.1.6 sub()

```
template<class Element >
std::enable_if< FFLAS::support_simd_add< Element >::value, void >::type sub (
    Element * T,
    const Element * TA,
    const Element * TB,
    size_t n) [inline]
```

#### 11.19.1.7 axpyp() [1/2]

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type axpyp
(
    const Field & F,
    const typename Field::Element a,
    typename Field::ConstElement_ptr X,
    typename Field::Element_ptr Y,
    const size_t n) [inline]
```

#### 11.19.1.8 axpyp() [2/2]

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type axpyp
(
    const Field & F,
    const typename Field::Element a,
    typename Field::ConstElement_ptr X,
    typename Field::Element_ptr Y,
    const size_t n,
    const size_t incX,
    const size_t incY) [inline]
```

**11.19.1.9 reduce()** [1/9]

```
template<class T >
std::enable_if<!std::is_integral< T >::value, T >::type reduce (
    T A,
    T B) [inline]
```

**11.19.1.10 reduce()** [2/9]

```
template<class T >
std::enable_if< std::is_integral< T >::value, T >::type reduce (
    T A,
    T B) [inline]
```

**11.19.1.11 reduce()** [3/9]

```
template<>
Givaro::Integer reduce (
    Givaro::Integer A,
    Givaro::Integer B) [inline]
```

**11.19.1.12 reduce()** [4/9]

```
float reduce (
    float A,
    float B,
    float invB,
    float min,
    float max) [inline]
```

**11.19.1.13 reduce()** [5/9]

```
double reduce (
    double A,
    double B,
    double invB,
    double min,
    double max) [inline]
```

**11.19.1.14 reduce()** [6/9]

```
int64_t reduce (
    int64_t A,
    int64_t p,
    double invp,
    double min,
    double max,
    int64_t pow50rem) [inline]
```



**11.19.1.15 reduce()** [7/9]

```
template<class Field >
Field::Element reduce (
    typename Field::Element A,
    HelperMod< Field, ElementCategories::MachineIntTag > & H) [inline]
```

**11.19.1.16 reduce()** [8/9]

```
template<class Field >
Field::Element reduce (
    typename Field::Element A,
    HelperMod< Field, ElementCategories::MachineFloatTag > & H) [inline]
```

**11.19.1.17 reduce()** [9/9]

```
template<class Field >
Field::Element reduce (
    typename Field::Element A,
    HelperMod< Field, ElementCategories::ArbitraryPrecIntTag > & H) [inline]
```

**11.19.1.18 modp()** [1/2]

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type modp (
    const Field & F,
    typename Field::ConstElement_ptr U,
    const size_t & n,
    typename Field::Element_ptr T) [inline]
```

**11.19.1.19 modp()** [2/2]

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type modp (
    const Field & F,
    typename Field::ConstElement_ptr U,
    const size_t & n,
    const size_t & incX,
    typename Field::Element_ptr T) [inline]
```

**11.19.1.20 scalp()** [1/3]

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp
(
    const Field & F,
    typename Field::Element_ptr T,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr U,
    const size_t n) [inline]
```

**11.19.1.21 scalp() [2/3]**

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp
(
    const Field & F,
    typename Field::Element_ptr T,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr U,
    const size_t n,
    const size_t & incX) [inline]
```

**11.19.1.22 scalp() [3/3]**

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp
(
    const Field & F,
    typename Field::Element_ptr T,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr U,
    const size_t n,
    const size_t & incX,
    const size_t & incY) [inline]
```

**11.20 FFLAS::vectorised::unswitch Namespace Reference****Functions**

- template<class Field >  
std::enable\_if<!FFLAS::support\_simd\_mod< typenameField::Element >::value &&FFLAS::support\_fast\_mod< typenameField::Element >::value, void >::type axypyp (const Field &F, const typename Field::Element a, typename Field::ConstElement\_ptr X, typename Field::Element\_ptr Y, const size\_t n, HelperMod< Field > &H)
- template<class Field >  
std::enable\_if< FFLAS::support\_fast\_mod< typenameField::Element >::value, void >::type axypyp (const Field &F, const typename Field::Element a, typename Field::ConstElement\_ptr X, typename Field::Element\_ptr Y, const size\_t n, const size\_t incX, const size\_t incY, HelperMod< Field > &H)
- template<class Field >  
std::enable\_if<!FFLAS::support\_simd\_mod< typenameField::Element >::value &&FFLAS::support\_fast\_mod< typenameField::Element >::value, void >::type modp (const Field &F, typename Field::ConstElement\_ptr U, const size\_t &n, typename Field::Element\_ptr T, HelperMod< Field > &H)
- template<class Field >  
std::enable\_if< FFLAS::support\_fast\_mod< typenameField::Element >::value, void >::type modp (const Field &F, typename Field::ConstElement\_ptr U, const size\_t &n, const size\_t &incX, typename Field::Element\_ptr T, HelperMod< Field > &H)
- template<class Field >  
std::enable\_if<!FFLAS::support\_simd\_mod< typenameField::Element >::value &&FFLAS::support\_fast\_mod< typenameField::Element >::value, void >::type scalp (const Field &F, typename Field::Element\_ptr T, const typename Field::Element alpha, typename Field::ConstElement\_ptr U, const size\_t n, HelperMod< Field > &H)

- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp`  
`(const Field &F, typename Field::Element_ptr T, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr U, const size_t n, const size_t &incX, HelperMod< Field > &H)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp`  
`(const Field &F, typename Field::Element_ptr T, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr U, const size_t n, const size_t &incX, const size_t &incY, HelperMod< Field >`  
`&H)`

## 11.20.1 Function Documentation

### 11.20.1.1 axypyp() [1/2]

```
template<class Field >
std::enable_if< !FFLAS::support_simd_mod< typenameField::Element >::value &&FFLAS::support_fast_mod<
typenameField::Element >::value, void >::type axypyp (
    const Field & F,
    const typename Field::Element a,
    typename Field::ConstElement_ptr X,
    typename Field::Element_ptr Y,
    const size_t n,
    HelperMod< Field > & H) [inline]
```

### 11.20.1.2 axypyp() [2/2]

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type axypyp
(
    const Field & F,
    const typename Field::Element a,
    typename Field::ConstElement_ptr X,
    typename Field::Element_ptr Y,
    const size_t n,
    const size_t incX,
    const size_t incY,
    HelperMod< Field > & H) [inline]
```

### 11.20.1.3 modp() [1/2]

```
template<class Field >
std::enable_if< !FFLAS::support_simd_mod< typenameField::Element >::value &&FFLAS::support_fast_mod<
typenameField::Element >::value, void >::type modp (
    const Field & F,
    typename Field::ConstElement_ptr U,
    const size_t & n,
    typename Field::Element_ptr T,
    HelperMod< Field > & H) [inline]
```

**11.20.1.4 modp() [2/2]**

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type modp (
    const Field & F,
    typename Field::ConstElement_ptr U,
    const size_t & n,
    const size_t & incX,
    typename Field::Element_ptr T,
    HelperMod< Field > & H) [inline]
```

**11.20.1.5 scalp() [1/3]**

```
template<class Field >
std::enable_if<!FFLAS::support_simd_mod< typenameField::Element >::value &&FFLAS::support_fast_mod<
typenameField::Element >::value, void >::type scalp (
    const Field & F,
    typename Field::Element_ptr T,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr U,
    const size_t n,
    HelperMod< Field > & H) [inline]
```

**11.20.1.6 scalp() [2/3]**

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp
(
    const Field & F,
    typename Field::Element_ptr T,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr U,
    const size_t n,
    const size_t & incX,
    HelperMod< Field > & H) [inline]
```

**11.20.1.7 scalp() [3/3]**

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp
(
    const Field & F,
    typename Field::Element_ptr T,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr U,
    const size_t n,
    const size_t & incX,
    const size_t & incY,
    HelperMod< Field > & H) [inline]
```

## 11.21 FFPACK Namespace Reference

Finite Field **PACK** Set of elimination based routines for dense linear algebra.

### Namespaces

- namespace [Protected](#)

### Data Structures

- class [callLUdivine\\_small](#)
- class [callLUdivine\\_small< double >](#)
- class [callLUdivine\\_small< float >](#)
- class [CharpolyFailed](#)
- class [CheckerImplem\\_charpoly](#)
- class [CheckerImplem\\_charpoly< Givaro::ZRing< Givaro::Integer >, Polynomial >](#)
- class [CheckerImplem\\_Det](#)
- class [CheckerImplem\\_invert](#)
- class [CheckerImplem\\_PLUQ](#)
- class [Failure](#)
  - A precondition failed.*
- struct [rns\\_double](#)
- struct [rns\\_double\\_elt](#)
- struct [rns\\_double\\_elt\\_cstptr](#)
- struct [rns\\_double\\_elt\\_ptr](#)
- struct [rns\\_double\\_extended](#)
- class [RNSInteger](#)
- class [RNSIntegerMod](#)
- class [rnsRandIter](#)

### Typedefs

- template<class [Field](#) >  
using [Checker\\_PLUQ](#) = [FFLAS::Checker\\_Empty<Field>](#)
- template<class [Field](#) >  
using [Checker\\_Det](#) = [FFLAS::Checker\\_Empty<Field>](#)
- template<class [Field](#) >  
using [Checker\\_invert](#) = [FFLAS::Checker\\_Empty<Field>](#)
- template<class [Field](#) , class [Polynomial](#) >  
using [Checker\\_charpoly](#) = [FFLAS::Checker\\_Empty<Field>](#)
- template<class [Field](#) >  
using [ForceCheck\\_PLUQ](#) = [CheckerImplem\\_PLUQ<Field>](#)
- template<class [Field](#) >  
using [ForceCheck\\_Det](#) = [CheckerImplem\\_Det<Field>](#)
- template<class [Field](#) >  
using [ForceCheck\\_invert](#) = [CheckerImplem\\_invert<Field>](#)
- template<class [Field](#) , class [Polynomial](#) >  
using [ForceCheck\\_charpoly](#) = [CheckerImplem\\_charpoly<Field,Polynomial>](#)

## Functions

- void [LAPACKPerm2MathPerm](#) (size\_t \*MathP, const size\_t \*LapackP, const size\_t N)  
*Conversion of a permutation from LAPACK format to Math format.*
- void [MathPerm2LAPACKPerm](#) (size\_t \*LapackP, const size\_t \*MathP, const size\_t N)  
*Conversion of a permutation from Maths format to LAPACK format.*
- template<class [Field](#) >  
void [applyP](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const [FFLAS::FFLAS\\_TRANSPOSE](#) Trans, const size\_t M, const size\_t ibeg, const size\_t iend, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t \*P)  
*Computes  $P1 \times \text{Diag}(I_R, P2)$  where  $P1$  is a LAPACK and  $P2$  a LAPACK permutation and store the result in  $P1$  as a LAPACK permutation.*
- template<class [Field](#) >  
void [applyP](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const [FFLAS::FFLAS\\_TRANSPOSE](#) Trans, const size\_t m, const size\_t ibeg, const size\_t iend, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t \*P, const [FFLAS::ParSeqHelper::Sequential](#) seq)
- template<class [Field](#) , class Cut , class Param >  
void [applyP](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const [FFLAS::FFLAS\\_TRANSPOSE](#) Trans, const size\_t m, const size\_t ibeg, const size\_t iend, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t \*P, const [FFLAS::ParSeqHelper::Parallel](#)< Cut, Param > par)
- template<class [Field](#) >  
void [MonotonicApplyP](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const [FFLAS::FFLAS\\_TRANSPOSE](#) Trans, const size\_t M, const size\_t ibeg, const size\_t iend, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t \*P, const size\_t R)  
*Apply a R-monotonically increasing permutation P, to the matrix A.*
- template<class [Field](#) >  
void [fgetrs](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t R, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t \*P, const size\_t \*Q, typename [Field::Element\\_ptr](#) B, const size\_t ldb, int \*info)  
*Solve the system  $AX = B$  or  $XA = B$ .*
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [fgetrs](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t NRHS, const size\_t R, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t \*P, const size\_t \*Q, typename [Field::Element\\_ptr](#) X, const size\_t ldx, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, int \*info)  
*Solve the system  $A X = B$  or  $X A = B$ .*
- template<class [Field](#) >  
size\_t [fgesv](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) B, const size\_t ldb, int \*info)  
*Square system solver.*
- template<class [Field](#) >  
size\_t [fgesv](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t NRHS, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) X, const size\_t ldx, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, int \*info)  
*Rectangular system solver.*
- template<class [Field](#) >  
void [ftrtri](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t threshold=[\\_\\_FFLASFFPACK\\_FTRTRI\\_THRESHOLD](#))  
*Compute the inverse of a triangular matrix.*
- template<class [Field](#) >  
void [trinv\\_left](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) L, const size\_t ldl, typename [Field::Element\\_ptr](#) X, const size\_t ldx)
- template<class [Field](#) >  
void [ftrtrm](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) side, const [FFLAS::FFLAS\\_DIAG](#) diag, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda)

*Compute the product of two triangular matrices of opposite shape.*

- `template<class Field >`  
`void ftrstr (const Field &F, const FFLAS::FFLAS_SIDE side, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diagA, const FFLAS::FFLAS_DIAG diagB, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, const size_t threshold=__FFLASFFPACK_FTRSTR_THRESHOLD)`

*Solve a triangular system with a triangular right hand side of the same shape.*

- `template<class Field >`  
`void ftrssyr2k (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diagA, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, const size_t threshold=__FFLASFFPACK_FTRSSYR2K_THRESHOLD)`

*Solve a triangular system in a symmetric sum: find B upper/lower triangular such that  $A^T B + B^T A = C$  where C is symmetric.*

- `template<class Field >`  
`bool fsytrf (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, const size_t threshold=__FFLASFFPACK_FSYTRF_THRESHOLD)`

*Triangular factorization of symmetric matrices.*

- `template<class Field >`  
`bool fsytrf (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, const FFLAS::ParSeqHelper::Sequential seq, const size_t threshold=__FFLASFFPACK_FSYTRF_THRESHOLD)`
- `template<class Field, class Cut, class Param >`  
`bool fsytrf (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, const FFLAS::ParSeqHelper::Parallel< Cut, Param > par, const size_t threshold=__FFLASFFPACK_FSYTRF_THRESHOLD)`
- `template<class Field >`  
`bool fsytrf_nonunit (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr D, const size_t incD, const size_t threshold=__FFLASFFPACK_FSYTRF_THRESHOLD)`

*Triangular factorization of symmetric matrices.*

- `template<class Field >`  
`size_t PLUQ (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q)`

*Compute a PLUQ factorization of the given matrix.*

- `template<class Field >`  
`size_t pPLUQ (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q)`
- `template<class Field >`  
`size_t PLUQ (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, const FFLAS::ParSeqHelper::Sequential &PShelper, size_t BCThreshold=__FFLASFFPACK_PLUQ_THRESHOLD)`
- `template<class Field, class Cut, class Param >`  
`size_t PLUQ (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, const FFLAS::ParSeqHelper::Parallel< Cut, Param > &PShelper)`
- `template<class Field >`  
`size_t LUdivine (const Field &F, const FFLAS::FFLAS_DIAG Diag, const FFLAS::FFLAS_TRANSPOSE trans, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const FFPACK_LU_TAG LuTag=FfpackSlabRecursive, const size_t cutoff=__FFLASFFPACK_LUDIVINE_THRESHOLD)`

*Compute the CUP or PLE factorization of the given matrix.*

- `template<class Field >`  
`size_t ColumnEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, bool transform=false, const FFPACK_LU_TAG LuTag=FfpackSlabRecursive)`

*Compute the Column Echelon form of the input matrix in-place.*

- `template<class Field >`  
`size_t pColumnEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, bool transform=false, size_t numthreads=0, const FFPACK_LU_TAG LuTag=FFpackTileRecursive)`
- `template<class Field , class PSHelper >`  
`size_t ColumnEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform, const FFPACK_LU_TAG LuTag, const PSHelper &psH)`
- `template<class Field >`  
`size_t RowEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform=false, const FFPACK_LU_TAG LuTag=FFpackSlabRecursive)`  
*Compute the Row Echelon form of the input matrix in-place.*
- `template<class Field >`  
`size_t pRowEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform=false, size_t numthreads=0, const FFPACK_LU_TAG LuTag=FFpackTileRecursive)`
- `template<class Field , class PSHelper >`  
`size_t RowEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform, const FFPACK_LU_TAG LuTag, const PSHelper &psH)`
- `template<class Field >`  
`size_t ReducedColumnEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform=false, const FFPACK_LU_TAG LuTag=FFpackSlabRecursive)`  
*Compute the Reduced Column Echelon form of the input matrix in-place.*
- `template<class Field >`  
`size_t pReducedColumnEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform=false, size_t numthreads=0, const FFPACK_LU_TAG LuTag=FFpackTileRecursive)`
- `template<class Field , class PSHelper >`  
`size_t ReducedColumnEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform, const FFPACK_LU_TAG LuTag, const PSHelper &psH)`
- `template<class Field >`  
`size_t ReducedRowEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform=false, const FFPACK_LU_TAG LuTag=FFpackSlabRecursive)`  
*Compute the Reduced Row Echelon form of the input matrix in-place.*
- `template<class Field >`  
`size_t pReducedRowEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform=false, size_t numthreads=0, const FFPACK_LU_TAG LuTag=FFpackTileRecursive)`
- `template<class Field , class PSHelper >`  
`size_t ReducedRowEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform, const FFPACK_LU_TAG LuTag, const PSHelper &psH)`
- `template<class Field >`  
`Field::Element_ptr Invert (const Field &F, const size_t M, typename Field::Element_ptr A, const size_t lda, int &>nullity)`  
*Invert the given matrix in place or computes its nullity if it is singular.*
- `template<class Field >`  
`Field::Element_ptr Invert (const Field &F, const size_t M, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr X, const size_t ldx, int &>nullity)`  
*Invert the given matrix or computes its nullity if it is singular.*



- `template<class Field >`  
`Field::Element_ptr Invert2` (const `Field` &F, const `size_t` M, typename `Field::Element_ptr` A, const `size_t` Ida, typename `Field::Element_ptr` X, const `size_t` Idx, int &>nullity)  
*Invert the given matrix or computes its nullity if it is singular.*
- `template<class PolRing >`  
`std::list< typename PolRing::Element > & CharPoly` (const `PolRing` &R, `std::list< typename PolRing::Element > &charp`, const `size_t` N, typename `PolRing::Domain_t::Element_ptr` A, const `size_t` Ida, typename `PolRing::Domain_t::RandIter` &G, const `FFPACK_CHARPOLY_TAG` CharpTag=FfpackAuto, const `size_t` degree=\_\_FFLASFFPACK\_ARITHPROG\_THRESHOLD)  
*Compute the characteristic polynomial of the matrix A.*
- `template<class PolRing >`  
`PolRing::Element & CharPoly` (const `PolRing` &R, typename `PolRing::Element` &charp, const `size_t` N, typename `PolRing::Domain_t::Element_ptr` A, const `size_t` Ida, typename `PolRing::Domain_t::RandIter` &G, const `FFPACK_CHARPOLY_TAG` CharpTag=FfpackAuto, const `size_t` degree=\_\_FFLASFFPACK\_ARITHPROG\_THRESHOLD)  
*Compute the characteristic polynomial of the matrix A.*
- `template<class PolRing >`  
`PolRing::Element & CharPoly` (const `PolRing` &R, typename `PolRing::Element` &charp, const `size_t` N, typename `PolRing::Domain_t::Element_ptr` A, const `size_t` Ida, const `FFPACK_CHARPOLY_TAG` CharpTag=FfpackAuto, const `size_t` degree=\_\_FFLASFFPACK\_ARITHPROG\_THRESHOLD)  
*Compute the characteristic polynomial of the matrix A.*
- `template<class Field , class Polynomial >`  
`Polynomial & MinPoly` (const `Field` &F, `Polynomial` &minP, const `size_t` N, typename `Field::ConstElement_ptr` A, const `size_t` Ida)  
*Compute the minimal polynomial of the matrix A.*
- `template<class Field , class Polynomial , class RandIter >`  
`Polynomial & MinPoly` (const `Field` &F, `Polynomial` &minP, const `size_t` N, typename `Field::ConstElement_ptr` A, const `size_t` Ida, `RandIter` &G)  
*Compute the minimal polynomial of the matrix A.*
- `template<class Field , class Polynomial >`  
`Polynomial & MatVecMinPoly` (const `Field` &F, `Polynomial` &minP, const `size_t` N, typename `Field::ConstElement_ptr` A, const `size_t` Ida, typename `Field::ConstElement_ptr` v, const `size_t` incv)  
*Compute the minimal polynomial of the matrix A and a vector v, namely the first linear dependency relation in the Krylov basis  $(v, Av, \dots, A^N v)$ .*
- `template<class Field >`  
`size_t Rank` (const `Field` &F, const `size_t` M, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` Ida)  
*Computes the rank of the given matrix using a PLUQ factorization.*
- `template<class Field >`  
`size_t pRank` (const `Field` &F, const `size_t` M, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` Ida, `size_t` numthreads=0)
- `template<class Field , class PSHelper >`  
`size_t Rank` (const `Field` &F, const `size_t` M, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` Ida, const `PSHelper` &psH)
- `template<class Field >`  
`bool IsSingular` (const `Field` &F, const `size_t` M, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` Ida)  
*Returns true if the given matrix is singular.*
- `template<class Field >`  
`Field::Element & Det` (const `Field` &F, typename `Field::Element` &det, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` Ida, `size_t` \*P=NULL, `size_t` \*Q=NULL)  
*Returns the determinant of the given square matrix.*
- `template<class Field >`  
`Field::Element & pDet` (const `Field` &F, typename `Field::Element` &det, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` Ida, `size_t` numthreads=0, `size_t` \*P=NULL, `size_t` \*Q=NULL)
- `template<class Field , class PSHelper >`  
`Field::Element & Det` (const `Field` &F, typename `Field::Element` &det, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` Ida, const `PSHelper` &psH, `size_t` \*P=NULL, `size_t` \*Q=NULL)

- `template<class Field >`  
`Field::Element_ptr Solve` (const `Field` &F, const `size_t` M, typename `Field::Element_ptr` A, const `size_t` Ida, typename `Field::Element_ptr` x, const `int` incx, typename `Field::ConstElement_ptr` b, const `int` incb)  
*Solves a linear system  $AX = b$  using PLUQ factorization.*
- `template<class Field , class PSHelper >`  
`Field::Element_ptr Solve` (const `Field` &F, const `size_t` M, typename `Field::Element_ptr` A, const `size_t` Ida, typename `Field::Element_ptr` x, const `int` incx, typename `Field::ConstElement_ptr` b, const `int` incb, PSHelper &psH)
- `template<class Field >`  
`Field::Element_ptr pSolve` (const `Field` &F, const `size_t` M, typename `Field::Element_ptr` A, const `size_t` Ida, typename `Field::Element_ptr` x, const `int` incx, typename `Field::ConstElement_ptr` b, const `int` incb, `size_t` numthreads=0)
- `template<class Field >`  
`*void RandomNullSpaceVector` (const `Field` &F, const `FFLAS::FFLAS_SIDE` Side, const `size_t` M, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` Ida, typename `Field::Element_ptr` X, const `size_t` incX)  
*Solve  $LX = B$  or  $XL = B$  in place.*
- `template<class Field >`  
`size_t NullSpaceBasis` (const `Field` &F, const `FFLAS::FFLAS_SIDE` Side, const `size_t` M, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` Ida, typename `Field::Element_ptr` &NS, `size_t` &Idn, `size_t` &NSdim)  
*Computes a basis of the Left/Right nullspace of the matrix A.*
- `template<class Field >`  
`size_t RowRankProfile` (const `Field` &F, const `size_t` M, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` Ida, `size_t` \*&rkprofile, const `FFPACK_LU_TAG` LuTag=`FfpackSlabRecursive`)  
*Computes the row rank profile of A.*
- `template<class Field >`  
`size_t pRowRankProfile` (const `Field` &F, const `size_t` M, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` Ida, `size_t` \*&rkprofile, `size_t` numthreads=0, const `FFPACK_LU_TAG` LuTag=`FfpackTileRecursive`)
- `template<class Field , class PSHelper >`  
`size_t RowRankProfile` (const `Field` &F, const `size_t` M, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` Ida, `size_t` \*&rkprofile, const `FFPACK_LU_TAG` LuTag, PSHelper &psH)
- `template<class Field >`  
`size_t ColumnRankProfile` (const `Field` &F, const `size_t` M, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` Ida, `size_t` \*&rkprofile, const `FFPACK_LU_TAG` LuTag=`FfpackSlabRecursive`)  
*Computes the column rank profile of A.*
- `template<class Field >`  
`size_t pColumnRankProfile` (const `Field` &F, const `size_t` M, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` Ida, `size_t` \*&rkprofile, `size_t` numthreads=0, const `FFPACK_LU_TAG` LuTag=`FfpackTileRecursive`)
- `template<class Field , class PSHelper >`  
`size_t ColumnRankProfile` (const `Field` &F, const `size_t` M, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` Ida, `size_t` \*&rkprofile, const `FFPACK_LU_TAG` LuTag, PSHelper &psH)
- `void RankProfileFromLU` (const `size_t` \*P, const `size_t` N, const `size_t` R, `size_t` \*&rkprofile, const `FFPACK_LU_TAG` LuTag)  
*Recovers the column/row rank profile from the permutation of an LU decomposition.*
- `size_t LeadingSubmatrixRankProfiles` (const `size_t` M, const `size_t` N, const `size_t` R, const `size_t` LSm, const `size_t` LSn, const `size_t` \*P, const `size_t` \*Q, `size_t` \*RRP, `size_t` \*CRP)  
*Recovers the row and column rank profiles of any leading submatrix from the PLUQ decomposition.*
- `template<class Field >`  
`size_t RowRankProfileSubmatrixIndices` (const `Field` &F, const `size_t` M, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` Ida, `size_t` \*&rowindices, `size_t` \*&colindices, `size_t` &R)  
*RowRankProfileSubmatrixIndices.*
- `template<class Field >`  
`size_t ColRankProfileSubmatrixIndices` (const `Field` &F, const `size_t` M, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` Ida, `size_t` \*&rowindices, `size_t` \*&colindices, `size_t` &R)  
*Computes the indices of the submatrix  $r \times r$  X of A whose columns correspond to the column rank profile of A.*

- `template<class Field >`  
`size_t RowRankProfileSubmatrix (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr &X, size_t &R)`  
*Computes the  $r \times r$  submatrix  $X$  of  $A$ , by picking the row rank profile rows of  $A$ .*
- `template<class Field >`  
`size_t ColRankProfileSubmatrix (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr &X, size_t &R)`  
*Compute the  $r \times r$  submatrix  $X$  of  $A$ , by picking the row rank profile rows of  $A$ .*
- `template<class Field >`  
`void getTriangular (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag, const size_t M, const size_t N, const size_t R, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr T, const size_t ldt, const bool OnlyNonZeroVectors=false)`  
*Extracts a triangular matrix from a compact storage  $A=L\backslash U$  of rank  $R$ .*
- `template<class Field >`  
`void getTriangular (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag, const size_t M, const size_t N, const size_t R, typename Field::Element_ptr A, const size_t lda)`  
*Cleans up a compact storage  $A=L\backslash U$  to reveal a triangular matrix of rank  $R$ .*
- `template<class Field >`  
`void getEchelonForm (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag, const size_t M, const size_t N, const size_t R, const size_t *P, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr T, const size_t ldt, const bool OnlyNonZeroVectors=false, const FFPACK_LU_TAG LuTag=FpackSlabRecursive)`  
*Extracts a matrix in echelon form from a compact storage  $A=L\backslash U$  of rank  $R$  obtained by RowEchelonForm or ColumnEchelonForm.*
- `template<class Field >`  
`void getEchelonForm (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag, const size_t M, const size_t N, const size_t R, const size_t *P, typename Field::Element_ptr A, const size_t lda, const FFPACK_LU_TAG LuTag=FpackSlabRecursive)`  
*Cleans up a compact storage  $A=L\backslash U$  obtained by RowEchelonForm or ColumnEchelonForm to reveal an echelon form of rank  $R$ .*
- `template<class Field >`  
`void getEchelonTransform (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag, const size_t M, const size_t N, const size_t R, const size_t *P, const size_t *Q, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr T, const size_t ldt, const FFPACK_LU_TAG LuTag=FpackSlabRecursive)`  
*Extracts a transformation matrix to echelon form from a compact storage  $A=L\backslash U$  of rank  $R$  obtained by RowEchelonForm or ColumnEchelonForm.*
- `template<class Field >`  
`void getReducedEchelonForm (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const size_t M, const size_t N, const size_t R, const size_t *P, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr T, const size_t ldt, const bool OnlyNonZeroVectors=false, const FFPACK_LU_TAG LuTag=FpackSlabRecursive)`  
*Extracts a matrix in echelon form from a compact storage  $A=L\backslash U$  of rank  $R$  obtained by ReducedRowEchelonForm or ReducedColumnEchelonForm with transform = true.*
- `template<class Field >`  
`void getReducedEchelonForm (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const size_t M, const size_t N, const size_t R, const size_t *P, typename Field::Element_ptr A, const size_t lda, const FFPACK_LU_TAG LuTag=FpackSlabRecursive)`  
*Cleans up a compact storage  $A=L\backslash U$  of rank  $R$  obtained by ReducedRowEchelonForm or ReducedColumnEchelonForm with transform = true.*
- `template<class Field >`  
`void getReducedEchelonTransform (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const size_t M, const size_t N, const size_t R, const size_t *P, const size_t *Q, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr T, const size_t ldt, const FFPACK_LU_TAG LuTag=FpackSlabRecursive)`  
*Extracts a transformation matrix to echelon form from a compact storage  $A=L\backslash U$  of rank  $R$  obtained by RowEchelonForm or ColumnEchelonForm.*

- void [PLUQtoEchelonPermutation](#) (const size\_t N, const size\_t R, const size\_t \*P, size\_t \*outPerm)  
*Auxiliary routine: determines the permutation that changes a PLUQ decomposition into a echelon form revealing PLUQ decomposition.*
- template<class [Field](#) >  
size\_t [LTBruhatGen](#) (const [Field](#) &Fi, const [FFLAS::FFLAS\\_DIAG](#) diag, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*P, size\_t \*Q)  
*LTBruhatGen Suppose A is Left Triangular Matrix This procedure computes the Bruhat Representation of A and return the rank of A.*
- template<class [Field](#) >  
void [getLTBruhatGen](#) (const [Field](#) &Fi, const size\_t N, const size\_t r, const size\_t \*P, const size\_t \*Q, typename [Field::Element\\_ptr](#) R, const size\_t ldr)  
*GetLTBruhatGen This procedure Computes the Rank Revealing Matrix based on the Bruhta representation of a Matrix.*
- template<class [Field](#) >  
void [getLTBruhatGen](#) (const [Field](#) &Fi, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const [FFLAS::FFLAS\\_DIAG](#) diag, const size\_t N, const size\_t r, const size\_t \*P, const size\_t \*Q, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) T, const size\_t ldt)  
*GetLTBruhatGen This procedure computes the matrix L or U f the Bruhat Representation Suppose that A is the bruhat representation of a matrix.*
- size\_t [LTQSorder](#) (const size\_t N, const size\_t r, const size\_t \*P, const size\_t \*Q)  
*LTQSorder This procedure computes the order of quasiseparability of a matrix.*
- template<class [Field](#) >  
size\_t [CompressToBlockBiDiagonal](#) (const [Field](#) &Fi, const [FFLAS::FFLAS\\_UPLO](#) Uplo, size\_t N, size\_t s, size\_t r, const size\_t \*P, const size\_t \*Q, typename [Field::Element\\_ptr](#) A, size\_t lda, typename [Field::Element\\_ptr](#) X, size\_t ldx, size\_t \*K, size\_t \*M, size\_t \*T)  
*CompressToBlockBiDiagonal This procedure compress a compact representation of a row echelon form or column echelon form.*
- template<class [Field](#) >  
void [ExpandBlockBiDiagonalToBruhat](#) (const [Field](#) &Fi, const [FFLAS::FFLAS\\_UPLO](#) Uplo, size\_t N, size\_t s, size\_t r, typename [Field::Element\\_ptr](#) A, size\_t lda, typename [Field::Element\\_ptr](#) X, size\_t ldx, size\_t NbBlocks, size\_t \*K, size\_t \*M, size\_t \*T)  
*ExpandBlockBiDiagonal This procedure expand a compact representation of a row echelon form or column echelon form.*
- void [Bruhat2EchelonPermutation](#) (size\_t N, size\_t R, const size\_t \*P, const size\_t \*Q, size\_t \*M)  
*Bruhat2EchelonPermutation (N,R,P,Q) Compute M such that LM or MU is in echelon form where L or U are factors of the Bruhat Rpresentation.*
- size\_t \* [TInverter](#) (size\_t \*T, size\_t r)
- template<class [Field](#) >  
void [ComputeRPermutation](#) (const [Field](#) &Fi, size\_t N, size\_t r, const size\_t \*P, const size\_t \*Q, size\_t \*R, size\_t \*MU, size\_t \*ML)
- template<class [Field](#) >  
void [productBruhatxTS](#) (const [Field](#) &Fi, size\_t N, size\_t s, size\_t r, const size\_t \*P, const size\_t \*Q, const typename [Field::Element\\_ptr](#) Xu, size\_t ldu, size\_t NbBlocksU, size\_t \*Ku, size\_t \*Tu, size\_t \*MU, const typename [Field::Element\\_ptr](#) XI, size\_t ldl, size\_t NbBlocksL, size\_t \*KI, size\_t \*TI, size\_t \*ML, typename [Field::Element\\_ptr](#) B, size\_t t, size\_t ldb, typename [Field::Element\\_ptr](#) C, size\_t ldc)  
*productBruhatxTS Comput the product between the CRE compact representation of a matrix A and B a tall matrix*
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [LQUPtoInverseOfFullRankMinor](#) (const [Field](#) &F, const size\_t rank, typename [Field::Element\\_ptr](#) A\_factors, const size\_t lda, const size\_t \*QtPointer, typename [Field::Element\\_ptr](#) X, const size\_t ldx)  
*LQUPtoInverseOfFullRankMinor.*
- template<class [Field](#) >  
void [RandomNullSpaceVector](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
*Solve  $LX = B$  or  $XL = B$  in place.*

- `template<class Field >`  
`void solveLB (const Field &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, const size_t R, typename Field::Element_ptr L, const size_t ldl, const size_t *Q, typename Field::Element_ptr B, const size_t ldb)`
- `template<class Field >`  
`void solveLB2 (const Field &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, const size_t R, typename Field::Element_ptr L, const size_t ldl, const size_t *Q, typename Field::Element_ptr B, const size_t ldb)`
- `size_t * Tinverter (const size_t *T, size_t r)`
- `template<class Field >`  
`void ComputeRPermutation (const Field &Fi, size_t N, size_t r, const size_t *P, const size_t *Q, size_t *R, const size_t *MU, const size_t *ML)`
- `template<class Field >`  
`Field::Element_ptr expandLCRE (const Field &Fi, size_t N, size_t s, size_t r, size_t *R, size_t i, typename Field::ConstElement_ptr Xu, size_t ldu, size_t NbBlocksU, const size_t *Ku, const size_t *Tuinv, typename Field::ConstElement_ptr XI, size_t ldl, size_t NbBlocksL, const size_t *KI, const size_t *Tlinv, typename Field::Element_ptr CRE, size_t ldcre)`  
*Expands an anti-diagonal block of a left triangular matrix from its compact Bruhat representation.*
- `template<class Field >`  
`void productBruhatxTS (const Field &Fi, size_t N, size_t s, size_t r, size_t t, const size_t *P, const size_t *Q, typename Field::ConstElement_ptr Xu, size_t ldu, size_t NbBlocksU, const size_t *Ku, const size_t *Tu, const size_t *MU, typename Field::ConstElement_ptr XI, size_t ldl, size_t NbBlocksL, const size_t *KI, const size_t *TI, const size_t *ML, typename Field::Element_ptr B, size_t ldb, const typename Field::Element beta, typename Field::Element_ptr D, size_t ldd)`  
*Compute the product of a left-triangular quasi-separable matrix A, represented by a compact Bruhat generator, with a dense rectangular matrix B:  $C \leftarrow A \times B + \text{beta}C$ .*
- `template<class Field , class Polynomial >`  
`std::list< Polynomial > & Danilevski (const Field &F, std::list< Polynomial > &charp, const size_t N, typename Field::Element_ptr A, const size_t lda)`
- `template<class Field >`  
`Field::Element_ptr buildMatrix (const Field &F, typename Field::ConstElement_ptr E, typename Field::ConstElement_ptr C, const size_t lda, const size_t *B, const size_t *T, const size_t me, const size_t mc, const size_t lambda, const size_t mu)`
- `FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr CharPoly (const FFPACK::RNSInteger< FFPACK::rns_double > &F, typename FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr charp, const size_t N, typename FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr A, const size_t lda, Givaro::ZRing< Givaro::Integer >::RandIter &G, const FFPACK_CHARPOLY_TAG CharpTag, size_t degree)`
- `template<> Givaro::Poly1Dom< Givaro::ZRing< Givaro::Integer > >::Element & CharPoly (const Givaro::Poly1Dom< Givaro::ZRing< Givaro::Integer > > &R, Givaro::Poly1Dom< Givaro::ZRing< Givaro::Integer > >::Element &charp, const size_t N, Givaro::Integer *A, const size_t lda, Givaro::ZRing< Givaro::Integer >::RandIter &G, const FFPACK_CHARPOLY_TAG CharpTag, size_t degree)`
- `template<class PSHelper >`  
`FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr & Det (const FFPACK::RNSInteger< FFPACK::rns_double > &F, typename FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr &det, const size_t N, typename FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr A, const size_t lda, const PSHelper &psH)`
- `template<class PSHelper >`  
`Givaro::Integer & Det (const Givaro::ZRing< Givaro::Integer > &F, Givaro::Integer &det, const size_t N, Givaro::Integer *A, const size_t lda, const PSHelper &psH, size_t *P, size_t *Q)`
- `template<class Field >`  
`bool fsytrf_BC_Crout (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr Din, const size_t incDin)`
- `template<class Field >`  
`size_t fsytrf_BC_RL (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr Din, const size_t incDin)`
- `template<class Field >`  
`size_t fsytrf_UP_RPM_BC_RL (const Field &F, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr Din, const size_t incDin, size_t *P)`

- `template<class Field >`  
`size_t fsytrf_LOW_RPM_BC_Crout (const Field &F, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr Dinv, const size_t incDinv, size_t *P)`
- `template<class Field >`  
`size_t fsytrf_UP_RPM_BC_Crout (const Field &F, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr Dinv, const size_t incDinv, size_t *P)`
- `template<class Field >`  
`size_t fsytrf_UP_RPM (const Field &F, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr Dinv, const size_t incDinv, size_t *P, size_t BCThreshold)`
- `template<class Field >`  
`bool fsytrf_nonunit (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr Dinv, const size_t incDinv, FFLAS::ParSeqHelper::Sequential seq, size_t threshold)`
- `template<class Field, class Cut, class Param >`  
`bool fsytrf_nonunit (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr Dinv, const size_t incDinv, FFLAS::ParSeqHelper::Parallel< Cut, Param > par, size_t threshold)`
- `template<class Field >`  
`size_t fsytrf_RPM (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t threshold)`
- `template<class Field >`  
`void getTridiagonal (const Field &F, const size_t N, const size_t R, typename Field::ConstElement_ptr A, const size_t lda, size_t *P, typename Field::Element_ptr T, const size_t ldt)`
- `template<class Field >`  
`size_t LUdivine_gauss (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, const FFPACK::FFPACK_LU_TAG LuTag)`
- `template<class Field >`  
`size_t LUdivine_small (const Field &F, const FFLAS::FFLAS_DIAG Diag, const FFLAS::FFLAS_TRANSPOSE trans, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, const FFPACK::FFPACK_LU_TAG LuTag)`
- `template<class Field >`  
`size_t LUdivine (const Field &F, const FFLAS::FFLAS_DIAG Diag, const FFLAS::FFLAS_TRANSPOSE trans, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, const FFPACK::FFPACK_LU_TAG LuTag, const size_t cutoff)`
- `template<> size_t LUdivine (const Givaro::Modular< Givaro::Integer > &F, const FFLAS::FFLAS_DIAG Diag, const FFLAS::FFLAS_TRANSPOSE trans, const size_t M, const size_t N, typename Givaro::Integer *A, const size_t lda, size_t *P, size_t *Q, const FFPACK::FFPACK_LU_TAG LuTag, const size_t cutoff)`
- `template<class Field >`  
`void MonotonicCompress (const Field &F, const FFLAS::FFLAS_SIDE Side, const size_t M, typename Field::Element_ptr A, const size_t lda, const size_t incA, const size_t *MathP, const size_t R, const size_t maxpiv, const size_t rowstomove, const std::vector< bool > &ispiv)`
- `template<class Field >`  
`void MonotonicCompressMorePivots (const Field &F, const FFLAS::FFLAS_SIDE Side, const size_t M, typename Field::Element_ptr A, const size_t lda, const size_t incA, const size_t *MathP, const size_t R, const size_t rowstomove, const size_t lenP)`
- `template<class Field >`  
`void MonotonicCompressCycles (const Field &F, const FFLAS::FFLAS_SIDE Side, const size_t M, typename Field::Element_ptr A, const size_t lda, const size_t incA, const size_t *MathP, const size_t lenP)`
- `template<class Field >`  
`void MonotonicExpand (const Field &F, const FFLAS::FFLAS_SIDE Side, const size_t M, typename Field::Element_ptr A, const size_t lda, const size_t incA, const size_t *MathP, const size_t R, const size_t maxpiv, const size_t rowstomove, const std::vector< bool > &ispiv)`
- `template<class Field >`  
`void applyP_block (const Field &F, const FFLAS::FFLAS_SIDE Side, const FFLAS::FFLAS_TRANSPOSE Trans, const size_t M, const size_t ibeg, const size_t iend, typename Field::Element_ptr A, const size_t lda, const size_t *P)`



- `template<class Field >`  
`void doApplyS (const Field &F, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr tmp, const size_t width, const size_t M2, const size_t R1, const size_t R2, const size_t R3, const size_t R4)`
- `template<class Field >`  
`void MatrixApplyS (const Field &F, typename Field::Element_ptr A, const size_t lda, const size_t width, const size_t M2, const size_t R1, const size_t R2, const size_t R3, const size_t R4)`
- `template<class Field >`  
`void MatrixApplyS (const Field &F, typename Field::Element_ptr A, const size_t lda, const size_t width, const size_t M2, const size_t R1, const size_t R2, const size_t R3, const size_t R4, const FFLAS::ParSeqHelper::Sequential seq)`
- `template<class Field, class Cut, class Param >`  
`void MatrixApplyS (const Field &F, typename Field::Element_ptr A, const size_t lda, const size_t width, const size_t M2, const size_t R1, const size_t R2, const size_t R3, const size_t R4, const FFLAS::ParSeqHelper::Parallel< Cut, Param > par)`
- `template<class T >`  
`void PermApplyS (T *A, const size_t lda, const size_t width, const size_t M2, const size_t R1, const size_t R2, const size_t R3, const size_t R4)`
- `template<class Field >`  
`void doApplyT (const Field &F, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr tmp, const size_t width, const size_t N2, const size_t R1, const size_t R2, const size_t R3, const size_t R4)`
- `template<class Field >`  
`void MatrixApplyT (const Field &F, typename Field::Element_ptr A, const size_t lda, const size_t width, const size_t N2, const size_t R1, const size_t R2, const size_t R3, const size_t R4)`
- `template<class Field >`  
`void MatrixApplyT (const Field &F, typename Field::Element_ptr A, const size_t lda, const size_t width, const size_t N2, const size_t R1, const size_t R2, const size_t R3, const size_t R4, const FFLAS::ParSeqHelper::Sequential seq)`
- `template<class Field, class Cut, class Param >`  
`void MatrixApplyT (const Field &F, typename Field::Element_ptr A, const size_t lda, const size_t width, const size_t N2, const size_t R1, const size_t R2, const size_t R3, const size_t R4, const FFLAS::ParSeqHelper::Parallel< Cut, Param > par)`
- `template<class T >`  
`void PermApplyT (T *A, const size_t lda, const size_t width, const size_t N2, const size_t R1, const size_t R2, const size_t R3, const size_t R4)`
- `void composePermutationsLLL (size_t *P1, const size_t *P2, const size_t R, const size_t N)`  
*Computes  $P1 \times \text{Diag}(I_R, P2)$  where  $P1$  is a LAPACK and  $P2$  a LAPACK permutation and store the result in  $P1$  as a LAPACK permutation.*
- `void composePermutationsLLM (size_t *MathP, const size_t *P1, const size_t *P2, const size_t R, const size_t N)`  
*Computes  $P1 \times \text{Diag}(I_R, P2)$  where  $P1$  is a LAPACK and  $P2$  a LAPACK permutation and store the result in  $MathP$  as a MathPermutation format.*
- `void composePermutationsMLM (size_t *MathP1, const size_t *P2, const size_t R, const size_t N)`  
*Computes  $MathP1 \times \text{Diag}(I_R, P2)$  where  $MathP1$  is a MathPermutation and  $P2$  a LAPACK permutation and store the result in  $MathP1$  as a MathPermutation format.*
- `void cyclic_shift_mathPerm (size_t *P, const size_t s)`
- `template<class Field >`  
`void cyclic_shift_row_col (const Field &F, typename Field::Element_ptr A, size_t m, size_t n, size_t lda)`
- `template<class Field >`  
`void cyclic_shift_row (const Field &F, typename Field::Element_ptr A, size_t m, size_t n, size_t lda)`
- `template<typename T >`  
`void cyclic_shift_row (const RNSIntegerMod< T > &F, typename T::Element_ptr A, size_t m, size_t n, size_t lda)`
- `template<class Field >`  
`void cyclic_shift_col (const Field &F, typename Field::Element_ptr A, size_t m, size_t n, size_t lda)`
- `template<typename T >`  
`void cyclic_shift_col (const RNSIntegerMod< T > &F, typename T::Element_ptr A, size_t m, size_t n, size_t lda)`

- `template<class Field >`  
`size_t PLUQ_basecaseV3 (const Field &Fi, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element *A, const size_t lda, size_t *P, size_t *Q)`
- `template<class Field >`  
`size_t PLUQ_basecaseV2 (const Field &Fi, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element *A, const size_t lda, size_t *P, size_t *Q)`
- `template<class Field >`  
`size_t PLUQ_basecaseCrout (const Field &Fi, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q)`
- `template<class Field >`  
`size_t _PLUQ (const Field &Fi, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, size_t BCThreshold)`
- `template<class Cut , class Param >`  
`size_t PLUQ (const Givaro::Modular< Givaro::Integer > &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Givaro::Integer *A, const size_t lda, size_t *P, size_t *Q, size_t BCThreshold, FFLAS::ParSeqHelper::Parallel< Cut, Param > &PSHelper)`
- `template<class Field >`  
`void threads_fgemm (const size_t m, const size_t n, const size_t r, int nbthreads, size_t *W1, size_t *W2, size_t *W3, size_t gamma)`
- `template<class Field >`  
`void threads_ftrsm (const size_t m, const size_t n, int nbthreads, size_t *t1, size_t *t2)`
- `template<class Field >`  
`size_t PLUQ (const Field &Fi, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, const FFLAS::ParSeqHelper::Parallel< FFLAS::CuttingStrategy::Recursive, FFLAS::StrategyParameter::Threads > &PSHelper)`
- `template<>` `rns_double_elt_ptr fflas_const_cast (rns_double_elt_cstptr x)`
- `template<>` `rns_double_elt_cstptr fflas_const_cast (rns_double_elt_ptr x)`
- `template<typename Base_t >`  
`void cyclic_shift_row_col (Base_t *A, size_t m, size_t n, size_t lda)`
- `template INST_OR_DECL` `void cyclic_shift_row (const FFLAS_FIELD< FFLAS_ELT > &F, FFLAS_ELT *A, size_t m, size_t n, size_t lda)`
- `template INST_OR_DECL` `void cyclic_shift_col (const FFLAS_FIELD< FFLAS_ELT > &F, FFLAS_ELT *A, size_t m, size_t n, size_t lda)`
- `template INST_OR_DECL` `void applyP (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_SIDE Side, const FFLAS::FFLAS_TRANSPOSE Trans, const size_t M, const size_t ibeg, const size_t iend, FFLAS_ELT *A, const size_t lda, const size_t *P)`
- `template INST_OR_DECL` `void fgetrs (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, const size_t R, FFLAS_ELT *A, const size_t lda, const size_t *P, const size_t *Q, FFLAS_ELT *B, const size_t ldb, int *info)`
- `template INST_OR_DECL` `FFLAS_ELT * fgetrs (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, const size_t NRHS, const size_t R, FFLAS_ELT *A, const size_t lda, const size_t *P, const size_t *Q, FFLAS_ELT *X, const size_t ldx, const FFLAS_ELT *B, const size_t ldb, int *info)`
- `template INST_OR_DECL` `size_t fgesv (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, FFLAS_ELT *A, const size_t lda, FFLAS_ELT *B, const size_t ldb, int *info)`
- `template INST_OR_DECL` `size_t fgesv (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, const size_t NRHS, FFLAS_ELT *A, const size_t lda, FFLAS_ELT *X, const size_t ldx, const FFLAS_ELT *B, const size_t ldb, int *info)`
- `template INST_OR_DECL` `void ftrtri (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG Diag, const size_t N, FFLAS_ELT *A, const size_t lda, const size_t threshold)`
- `template INST_OR_DECL` `void trinv_left (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t N, const FFLAS_ELT *L, const size_t ldl, FFLAS_ELT *X, const size_t ldx)`
- `template INST_OR_DECL` `void ftrtrm (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_SIDE side, const FFLAS::FFLAS_DIAG diag, const size_t N, FFLAS_ELT *A, const size_t lda)`



- template `INST_OR_DECL` `size_t PLUQ` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS::FFLAS_DIAG` Diag, const `size_t M`, const `size_t N`, `FFLAS_ELT *A`, const `size_t lda`, `size_t *P`, `size_t *Q`)
- template `INST_OR_DECL` `size_t LUdivine` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS::FFLAS_DIAG` Diag, const `FFLAS::FFLAS_TRANSPOSE` trans, const `size_t M`, const `size_t N`, `FFLAS_ELT *A`, const `size_t lda`, `size_t *P`, `size_t *Qt`, const `FFPACK_LU_TAG` LuTag, const `size_t cutoff`)
- template `INST_OR_DECL` `size_t LUdivine_small` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS::FFLAS_DIAG` Diag, const `FFLAS::FFLAS_TRANSPOSE` trans, const `size_t M`, const `size_t N`, `FFLAS_ELT *A`, const `size_t lda`, `size_t *P`, `size_t *Q`, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` `size_t LUdivine_gauss` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS::FFLAS_DIAG` Diag, const `size_t M`, const `size_t N`, `FFLAS_ELT *A`, const `size_t lda`, `size_t *P`, `size_t *Q`, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` `size_t RowEchelonForm` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t M`, const `size_t N`, `FFLAS_ELT *A`, const `size_t lda`, `size_t *P`, `size_t *Qt`, const `bool transform`, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` `size_t ReducedRowEchelonForm` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t M`, const `size_t N`, `FFLAS_ELT *A`, const `size_t lda`, `size_t *P`, `size_t *Qt`, const `bool transform`, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` `size_t ColumnEchelonForm` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t M`, const `size_t N`, `FFLAS_ELT *A`, const `size_t lda`, `size_t *P`, `size_t *Qt`, const `bool transform`, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` `size_t ReducedColumnEchelonForm` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t M`, const `size_t N`, `FFLAS_ELT *A`, const `size_t lda`, `size_t *P`, `size_t *Qt`, const `bool transform`, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` `FFLAS_ELT * Invert` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t M`, `FFLAS_ELT *A`, const `size_t lda`, `int &nullity`)
- template `INST_OR_DECL` `FFLAS_ELT * Invert` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t M`, const `FFLAS_ELT *A`, const `size_t lda`, `FFLAS_ELT *X`, const `size_t ldx`, `int &nullity`)
- template `INST_OR_DECL` `FFLAS_ELT * Invert2` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t M`, `FFLAS_ELT *A`, const `size_t lda`, `FFLAS_ELT *X`, const `size_t ldx`, `int &nullity`)
- template `INST_OR_DECL` `std::list< Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element > & CharPoly` (const `Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > > &R`, `std::list< Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element > &charp`, const `size_t N`, `FFLAS_ELT *A`, const `size_t lda`, `FFLAS_FIELD< FFLAS_ELT >::RandIter &G`, const `FFPACK_CHARPOLY_TAG` CharpTag, const `size_t degree`)
- template `INST_OR_DECL` `Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element & CharPoly` (const `Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > > &R`, `Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element &charp`, const `size_t N`, `FFLAS_ELT *A`, const `size_t lda`, `FFLAS_FIELD< FFLAS_ELT >::RandIter &G`, const `FFPACK_CHARPOLY_TAG` CharpTag, const `size_t degree`)
- template `INST_OR_DECL` `Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element & CharPoly` (const `Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > > &R`, `Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element &charp`, const `size_t N`, `FFLAS_ELT *A`, const `size_t lda`, const `FFPACK_CHARPOLY_TAG` CharpTag, const `size_t degree`)
- template `INST_OR_DECL` `std::vector< FFLAS_ELT > & MinPoly` (const `FFLAS_FIELD< FFLAS_ELT >` &F, `std::vector< FFLAS_ELT > &minP`, const `size_t N`, const `FFLAS_ELT *A`, const `size_t lda`, `FFLAS_FIELD< FFLAS_ELT >::RandIter &G`)
- template `INST_OR_DECL` `std::vector< FFLAS_ELT > & MinPoly` (const `FFLAS_FIELD< FFLAS_ELT >` &F, `std::vector< FFLAS_ELT > &minP`, const `size_t N`, const `FFLAS_ELT *A`, const `size_t lda`)
- template `INST_OR_DECL` `std::vector< FFLAS_ELT > & MatVecMinPoly` (const `FFLAS_FIELD< FFLAS_ELT >` &F, `std::vector< FFLAS_ELT > &minP`, const `size_t N`, const `FFLAS_ELT *A`, const `size_t lda`, const `FFLAS_ELT *V`, const `size_t incv`)
- template `INST_OR_DECL` `size_t KrylovElim` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t M`, const `size_t N`, `FFLAS_ELT *A`, const `size_t lda`, `size_t *P`, `size_t *Q`, const `size_t deg`, `size_t *iterates`, `size_t *inviterates`, const `size_t maxit`, `size_t virt`)
- template `INST_OR_DECL` `size_t SpecRankProfile` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t M`, const `size_t N`, `FFLAS_ELT *A`, const `size_t lda`, const `size_t deg`, `size_t *rankProfile`)
- template `INST_OR_DECL` `size_t Rank` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t M`, const `size_t N`, `FFLAS_ELT *A`, const `size_t lda`)

- template `INST_OR_DECL` bool `IsSingular` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, `FFLAS_ELT` \*A, const size\_t lda)
- template `INST_OR_DECL` `FFLAS_ELT` & `Det` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, `FFLAS_ELT` &det, const size\_t N, `FFLAS_ELT` \*A, const size\_t lda, size\_t \*P, size\_t \*Q)
- template `INST_OR_DECL` `FFLAS_ELT` & `Det` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, `FFLAS_ELT` &det, const size\_t N, `FFLAS_ELT` \*A, const size\_t lda, const `FFLAS::ParSeqHelper::Parallel`< `FFLAS::CuttingStrategy::Recursive`, `FFLAS::StrategyParameter::Threads` > &parH, size\_t \*P, size\_t \*Q)
- template `INST_OR_DECL` void `solveLB` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS::FFLAS_SIDE` Side, const size\_t M, const size\_t N, const size\_t R, `FFLAS_ELT` \*L, const size\_t ldl, const size\_t \*Q, `FFLAS_ELT` \*B, const size\_t ldb)
- template `INST_OR_DECL` void `solveLB2` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS::FFLAS_SIDE` Side, const size\_t M, const size\_t N, const size\_t R, `FFLAS_ELT` \*L, const size\_t ldl, const size\_t \*Q, `FFLAS_ELT` \*B, const size\_t ldb)
- template `INST_OR_DECL` void `RandomNullSpaceVector` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS::FFLAS_SIDE` Side, const size\_t M, const size\_t N, `FFLAS_ELT` \*A, const size\_t lda, `FFLAS_ELT` \*X, const size\_t incX)
- template `INST_OR_DECL` size\_t `NullSpaceBasis` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS::FFLAS_SIDE` Side, const size\_t M, const size\_t N, `FFLAS_ELT` \*A, const size\_t lda, `FFLAS_ELT` \*&NS, size\_t &ldn, size\_t &NSdim)
- template `INST_OR_DECL` size\_t `RowRankProfile` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, `FFLAS_ELT` \*A, const size\_t lda, size\_t \*&rkprofile, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` size\_t `ColumnRankProfile` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, `FFLAS_ELT` \*A, const size\_t lda, size\_t \*&rkprofile, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` size\_t `RowRankProfileSubmatrixIndices` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, `FFLAS_ELT` \*A, const size\_t lda, size\_t \*&rowindices, size\_t \*&colindices, size\_t &R)
- template `INST_OR_DECL` size\_t `ColRankProfileSubmatrixIndices` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, `FFLAS_ELT` \*A, const size\_t lda, size\_t \*&rowindices, size\_t \*&colindices, size\_t &R)
- template `INST_OR_DECL` size\_t `RowRankProfileSubmatrix` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, `FFLAS_ELT` \*A, const size\_t lda, `FFLAS_ELT` \*&X, size\_t &R)
- template `INST_OR_DECL` size\_t `ColRankProfileSubmatrix` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, `FFLAS_ELT` \*A, const size\_t lda, `FFLAS_ELT` \*&X, size\_t &R)
- template `INST_OR_DECL` void `getTriangular`< `FFLAS_FIELD`< `FFLAS_ELT` > > (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS::FFLAS_UPLO` Uplo, const `FFLAS::FFLAS_DIAG` diag, const size\_t M, const size\_t N, const size\_t R, const `FFLAS_ELT` \*A, const size\_t lda, `FFLAS_ELT` \*T, const size\_t ldt, const bool OnlyNonZeroVectors)
- template `INST_OR_DECL` void `getTriangular`< `FFLAS_FIELD`< `FFLAS_ELT` > > (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS::FFLAS_UPLO` Uplo, const `FFLAS::FFLAS_DIAG` diag, const size\_t M, const size\_t N, const size\_t R, `FFLAS_ELT` \*A, const size\_t lda)
- template `INST_OR_DECL` void `getEchelonForm`< `FFLAS_FIELD`< `FFLAS_ELT` > > (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS::FFLAS_UPLO` Uplo, const `FFLAS::FFLAS_DIAG` diag, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const `FFLAS_ELT` \*A, const size\_t lda, `FFLAS_ELT` \*T, const size\_t ldt, const bool OnlyNonZeroVectors, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` void `getEchelonForm`< `FFLAS_FIELD`< `FFLAS_ELT` > > (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS::FFLAS_UPLO` Uplo, const `FFLAS::FFLAS_DIAG` diag, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, `FFLAS_ELT` \*A, const size\_t lda, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` void `getEchelonTransform`< `FFLAS_FIELD`< `FFLAS_ELT` > > (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS::FFLAS_UPLO` Uplo, const `FFLAS::FFLAS_DIAG` diag, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const size\_t \*Q, const `FFLAS_ELT` \*A, const size\_t lda, `FFLAS_ELT` \*T, const size\_t ldt, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` void `getReducedEchelonForm`< `FFLAS_FIELD`< `FFLAS_ELT` > > (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS::FFLAS_UPLO` Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const `FFLAS_ELT` \*A, const size\_t lda, `FFLAS_ELT` \*T, const size\_t ldt, const bool OnlyNonZeroVectors, const `FFPACK_LU_TAG` LuTag)

- template INST\_OR\_DECL void [getReducedEchelonForm](#)< FFLAS\_FIELD< FFLAS\_ELT > > (const FFLAS\_FIELD< FFLAS\_ELT > &F, const FFLAS::FFLAS\_UPLO Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, FFLAS\_ELT \*A, const size\_t lda, const FFPACK\_LU\_TAG LuTag)
- template INST\_OR\_DECL void [getReducedEchelonTransform](#)< FFLAS\_FIELD< FFLAS\_ELT > > (const FFLAS\_FIELD< FFLAS\_ELT > &F, const FFLAS::FFLAS\_UPLO Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const size\_t \*Q, const FFLAS\_ELT \*A, const size\_t lda, FFLAS\_ELT \*T, const size\_t ldt, const FFPACK\_LU\_TAG LuTag)
- template INST\_OR\_DECL FFLAS\_ELT \* [LQUPtoInverseOfFullRankMinor](#) (const FFLAS\_FIELD< FFLAS\_ELT > &F, const size\_t rank, FFLAS\_ELT \*A\_factors, const size\_t lda, const size\_t \*QtPointer, FFLAS\_ELT \*X, const size\_t ldx)
- template<class T, class CT = const T>  
T [fflas\\_const\\_cast](#) (CT x)
- [Failure](#) & [failure](#) ()
- template<class T >  
bool [isOdd](#) (const T &a)
- bool [isOdd](#) (const float &a)
- bool [isOdd](#) (const double &a)
- template<class Field, class Randlter >  
[Field::Element\\_ptr NonZeroRandomMatrix](#) (const Field &F, size\_t m, size\_t n, typename Field::Element\_ptr A, size\_t lda, Randlter &G)  
*Random non-zero Matrix.*
- template<class Field, class Randlter >  
[Field::Element\\_ptr NonZeroRandomMatrix](#) (const Field &F, size\_t m, size\_t n, typename Field::Element\_ptr A, size\_t lda)  
*Random non-zero Matrix.*
- template<class Field, class Randlter >  
[Field::Element\\_ptr RandomMatrix](#) (const Field &F, size\_t m, size\_t n, typename Field::Element\_ptr A, size\_t lda, Randlter &G)  
*Random Matrix.*
- template<class Field >  
[Field::Element\\_ptr RandomMatrix](#) (const Field &F, size\_t m, size\_t n, typename Field::Element\_ptr A, size\_t lda)  
*Random Matrix.*
- template<class Field, class Randlter >  
[Field::Element\\_ptr RandomTriangularMatrix](#) (const Field &F, size\_t m, size\_t n, const FFLAS::FFLAS\_UPLO UpLo, const FFLAS::FFLAS\_DIAG Diag, bool nonsingular, typename Field::Element\_ptr A, size\_t lda, Randlter &G)  
*Random Triangular Matrix.*
- template<class Field >  
[Field::Element\\_ptr RandomTriangularMatrix](#) (const Field &F, size\_t m, size\_t n, const FFLAS::FFLAS\_UPLO UpLo, const FFLAS::FFLAS\_DIAG Diag, bool nonsingular, typename Field::Element\_ptr A, size\_t lda)  
*Random Triangular Matrix.*
- size\_t [RandInt](#) (size\_t a, size\_t b)
- template<class Field, class Randlter >  
[Field::Element\\_ptr RandomSymmetricMatrix](#) (const Field &F, size\_t n, bool nonsingular, typename Field::Element\_ptr A, size\_t lda, Randlter &G)  
*Random Symmetric Matrix.*
- template<class Field, class Randlter >  
[Field::Element\\_ptr RandomMatrixWithRank](#) (const Field &F, size\_t m, size\_t n, size\_t r, typename Field::Element\_ptr A, size\_t lda, Randlter &G)  
*Random Matrix with prescribed rank.*
- template<class Field >  
[Field::Element\\_ptr RandomMatrixWithRank](#) (const Field &F, size\_t m, size\_t n, size\_t r, typename Field::Element\_ptr A, size\_t lda)  
*Random Matrix with prescribed rank.*

- `size_t * RandomIndexSubset` (`size_t N`, `size_t R`, `size_t *P`)  
*Pick uniformly at random a sequence of  $R$  distinct elements from the set  $\{0, \dots, N - 1\}$  using Knuth's shuffle.*
- `size_t * RandomPermutation` (`size_t N`, `size_t *P`)  
*Pick uniformly at random a permutation of size  $N$  stored in LAPACK format using Knuth's shuffle.*
- `void RandomRankProfileMatrix` (`size_t M`, `size_t N`, `size_t R`, `size_t *rows`, `size_t *cols`)  
*Pick uniformly at random an  $R$ -subpermutation of dimension  $M \times N$  : a matrix with only  $R$  non-zeros equal to one, in a random rook placement.*
- `void swapval` (`size_t k`, `size_t N`, `size_t *P`, `size_t val`)
- `void RandomSymmetricRankProfileMatrix` (`size_t N`, `size_t R`, `size_t *rows`, `size_t *cols`)  
*Pick uniformly at random a symmetric  $R$ -subpermutation of dimension  $N \times N$  : a symmetric matrix with only  $R$  non-zeros, all equal to one, in a random rook placement.*
- `void RandomLTQSRankProfileMatrix` (`size_t n`, `size_t r`, `size_t t`, `size_t *rows`, `size_t *cols`)
- `template<class Field, class RandIter >`  
`Field::Element_ptr RandomMatrixWithRankandRPM` (`const Field &F`, `size_t M`, `size_t N`, `size_t R`, `typename Field::Element_ptr A`, `size_t lda`, `const size_t *RRP`, `const size_t *CRP`, `RandIter &G`)  
*Random Matrix with prescribed rank and rank profile matrix Creates an  $m \times n$  matrix with random entries and rank  $r$ .*
- `template<class Field >`  
`Field::Element_ptr RandomMatrixWithRankandRPM` (`const Field &F`, `size_t M`, `size_t N`, `size_t R`, `typename Field::Element_ptr A`, `size_t lda`, `const size_t *RRP`, `const size_t *CRP`)  
*Random Matrix with prescribed rank and rank profile matrix Creates an  $m \times n$  matrix with random entries and rank  $r$ .*
- `template<class Field, class RandIter >`  
`Field::Element_ptr RandomSymmetricMatrixWithRankandRPM` (`const Field &F`, `size_t N`, `size_t R`, `typename Field::Element_ptr A`, `size_t lda`, `const size_t *RRP`, `const size_t *CRP`, `RandIter &G`)  
*Random Symmetric Matrix with prescribed rank and rank profile matrix Creates an  $n \times n$  symmetric matrix with random entries and rank  $r$ .*
- `template<class Field >`  
`Field::Element_ptr RandomSymmetricMatrixWithRankandRPM` (`const Field &F`, `size_t M`, `size_t N`, `size_t R`, `typename Field::Element_ptr A`, `size_t lda`, `const size_t *RRP`, `const size_t *CRP`)  
*Random Symmetric Matrix with prescribed rank and rank profile matrix Creates an  $n \times n$  symmetric matrix with random entries and rank  $r$ .*
- `template<class Field, class RandIter >`  
`Field::Element_ptr RandomMatrixWithRankandRandomRPM` (`const Field &F`, `size_t M`, `size_t N`, `size_t R`, `typename Field::Element_ptr A`, `size_t lda`, `RandIter &G`)  
*Random Matrix with prescribed rank, with random rank profile matrix Creates an  $m \times n$  matrix with random entries, rank  $r$  and with a rank profile matrix chosen uniformly at random.*
- `template<class Field >`  
`Field::Element_ptr RandomMatrixWithRankandRandomRPM` (`const Field &F`, `size_t M`, `size_t N`, `size_t R`, `typename Field::Element_ptr A`, `size_t lda`)  
*Random Matrix with prescribed rank, with random rank profile matrix Creates an  $m \times n$  matrix with random entries, rank  $r$  and with a rank profile matrix chosen uniformly at random.*
- `template<class Field, class RandIter >`  
`Field::Element_ptr RandomSymmetricMatrixWithRankandRandomRPM` (`const Field &F`, `size_t N`, `size_t R`, `typename Field::Element_ptr A`, `size_t lda`, `RandIter &G`)  
*Random Symmetric Matrix with prescribed rank, with random rank profile matrix Creates an  $n \times n$  matrix with random entries, rank  $r$  and with a rank profile matrix chosen uniformly at random.*
- `template<class Field >`  
`Field::Element_ptr RandomSymmetricMatrixWithRankandRandomRPM` (`const Field &F`, `size_t N`, `size_t R`, `typename Field::Element_ptr A`, `size_t lda`)  
*Random Symmetric Matrix with prescribed rank, with random rank profile matrix Creates an  $n \times n$  matrix with random entries, rank  $r$  and with a rank profile matrix chosen uniformly at random.*
- `template<class Field >`  
`Field::Element_ptr RandomMatrixWithDet` (`const Field &F`, `size_t n`, `const typename Field::Element d`, `typename Field::Element_ptr A`, `size_t lda`)  
*Random Matrix with prescribed det.*

- `template<class Field , class Randlter >`  
`Field::Element_ptr RandomMatrixWithDet` (const `Field` &F, `size_t` n, const typename `Field::Element` d, type-  
name `Field::Element_ptr` A, `size_t` lda, `Randlter` &G)  
*Random Matrix with prescribed det.*
- `template<class Field , class Randlter >`  
`Field::Element_ptr RandomLTQSMatWithRankandQSorder` (`Field` &F, `size_t` n, `size_t` r, `size_t` t, typename  
`Field::Element_ptr` A, `size_t` lda, `Randlter` &G)
- `template<typename Field >`  
`Field * chooseField` (`Givaro::Integer` q, `uint64_t` b, `uint64_t` seed)
- `template<> Givaro::ZRing< int32_t > * chooseField< Givaro::ZRing< int32_t > >` (`Givaro::Integer` q,  
`uint64_t` b, `uint64_t` seed)
- `template<> Givaro::ZRing< int64_t > * chooseField< Givaro::ZRing< int64_t > >` (`Givaro::Integer` q,  
`uint64_t` b, `uint64_t` seed)
- `template<> Givaro::ZRing< float > * chooseField< Givaro::ZRing< float > >` (`Givaro::Integer` q, `uint64_t`  
b, `uint64_t` seed)
- `template<> Givaro::ZRing< double > * chooseField< Givaro::ZRing< double > >` (`Givaro::Integer` q,  
`uint64_t` b, `uint64_t` seed)

### 11.21.1 Detailed Description

**Finite Field PACK** Set of elimination based routines for dense linear algebra.

This namespace enlarges the set of BLAS routines of the class `FFLAS`, with higher level routines based on elimination.

### 11.21.2 Typedef Documentation

#### 11.21.2.1 Checker\_PLUQ

```
template<class Field >
using Checker_PLUQ = FFLAS::Checker_Empty<Field>
```

#### 11.21.2.2 Checker\_Det

```
template<class Field >
using Checker_Det = FFLAS::Checker_Empty<Field>
```

#### 11.21.2.3 Checker\_invert

```
template<class Field >
using Checker_invert = FFLAS::Checker_Empty<Field>
```

#### 11.21.2.4 Checker\_charpoly

```
template<class Field , class Polynomial >
using Checker_charpoly = FFLAS::Checker_Empty<Field>
```

### 11.21.2.5 ForceCheck\_PLUQ

```
template<class Field >
using ForceCheck_PLUQ = CheckerImplem_PLUQ<Field>
```

### 11.21.2.6 ForceCheck\_Det

```
template<class Field >
using ForceCheck_Det = CheckerImplem_Det<Field>
```

### 11.21.2.7 ForceCheck\_invert

```
template<class Field >
using ForceCheck_invert = CheckerImplem_invert<Field>
```

### 11.21.2.8 ForceCheck\_charpoly

```
template<class Field , class Polynomial >
using ForceCheck_charpoly = CheckerImplem_charpoly<Field,Polynomial>
```

## 11.21.3 Function Documentation

### 11.21.3.1 LAPACKPerm2MathPerm()

```
void LAPACKPerm2MathPerm (
    size_t * MathP,
    const size_t * LapackP,
    const size_t N) [inline]
```

Conversion of a permutation from LAPACK format to Math format.

### 11.21.3.2 MathPerm2LAPACKPerm()

```
void MathPerm2LAPACKPerm (
    size_t * LapackP,
    const size_t * MathP,
    const size_t N) [inline]
```

Conversion of a permutation from Maths format to LAPACK format.

### 11.21.3.3 applyP() [1/4]

```
template<class Field >
void applyP (
    const Field & F,
    const FFLAS::FFLAS_SIDE Side,
    const FFLAS::FFLAS_TRANSPOSE Trans,
    const size_t M,
    const size_t ibeg,
    const size_t iend,
    typename Field::Element_ptr A,
    const size_t lda,
    const size_t * P) [inline]
```

Computes  $P1 \times \text{Diag}(I\_R, P2)$  where  $P1$  is a LAPACK and  $P2$  a LAPACK permutation and store the result in  $P1$  as a LAPACK permutation.

## Parameters

|                |           |                                  |
|----------------|-----------|----------------------------------|
| <i>in, out</i> | <i>P1</i> | a LAPACK permutation of size N   |
|                | <i>P2</i> | a LAPACK permutation of size N-R |

Applies a permutation P to the matrix A. Apply a permutation P, stored in the LAPACK format (a sequence of transpositions) between indices *ibeg* and *iend* of P to (*iend-ibeg*) vectors of size M stored in A (as column for NoTrans and rows for Trans). Side==FFLAS::FflasLeft for row permutation Side==FFLAS::FflasRight for a column permutation Trans==FFLAS::FflasTrans for the inverse permutation of P

## Parameters

|              |                                                                                                 |
|--------------|-------------------------------------------------------------------------------------------------|
| <i>F</i>     | base field                                                                                      |
| <i>Side</i>  | decides if rows (FflasLeft) or columns (FflasRight) are permuted                                |
| <i>Trans</i> | decides if the matrix is seen as columns (FflasTrans) or rows (FflasNoTrans)                    |
| <i>M</i>     | size of the elements to permute                                                                 |
| <i>ibeg</i>  | first index to consider in P                                                                    |
| <i>iend</i>  | last index to consider in P                                                                     |
| <i>A</i>     | input matrix                                                                                    |
| <i>lda</i>   | leading dimension of A                                                                          |
| <i>P</i>     | permutation in LAPACK format                                                                    |
| <i>psh</i>   | (optional): a sequential or parallel helper, to choose between sequential or parallel execution |

## Warning

not sure the submatrix is still a permutation and the one we expect in all cases... examples for *iend*=2, *ibeg*=1 and *P*=[2,2,2]

## 11.21.3.4 applyP() [2/4]

```
template<class Field >
void applyP (
    const Field & F,
    const FFLAS::FFLAS_SIDE Side,
    const FFLAS::FFLAS_TRANSPOSE Trans,
    const size_t m,
    const size_t ibeg,
    const size_t iend,
    typename Field::Element_ptr A,
    const size_t lda,
    const size_t * P,
    const FFLAS::ParSeqHelper::Sequential seq) [inline]
```

## 11.21.3.5 applyP() [3/4]

```
template<class Field , class Cut , class Param >
void applyP (
    const Field & F,
    const FFLAS::FFLAS_SIDE Side,
```

```

const FFLAS::FFLAS_TRANSPOSE Trans,
const size_t m,
const size_t ibeg,
const size_t iend,
typename Field::Element_ptr A,
const size_t lda,
const size_t * P,
const FFLAS::ParSeqHelper::Parallel< Cut, Param > par) [inline]

```

### 11.21.3.6 MonotonicApplyP()

```

template<class Field >
void MonotonicApplyP (
    const Field & F,
    const FFLAS::FFLAS_SIDE Side,
    const FFLAS::FFLAS_TRANSPOSE Trans,
    const size_t M,
    const size_t ibeg,
    const size_t iend,
    typename Field::Element_ptr A,
    const size_t lda,
    const size_t * P,
    const size_t R) [inline]

```

Apply a R-monotonically increasing permutation P, to the matrix A.

MonotonicApplyP Apply a permutation defined by the first R entries of the vector P (the pivots).

The permutation represented by P is defined as follows:

- the first R values of P is a LAPACK representation (a sequence of transpositions)
- the remaining iend-ibeg-R values of the permutation are in a monotonically increasing progression Side==FFLAS::FflasLeft for row permutation Side==FFLAS::FflasRight for a column permutation Trans==FFLAS::FflasTrans for the inverse permutation of P

#### Parameters

|              |                                                                       |
|--------------|-----------------------------------------------------------------------|
| <i>F</i>     | base field                                                            |
| <i>Side</i>  | selects if it is a row (FflasLeft) or column (FflasRight) permutation |
| <i>Trans</i> | inverse permutation (FflasTrans/NoTrans)                              |
| <i>M</i>     |                                                                       |
| <i>ibeg</i>  |                                                                       |
| <i>iend</i>  |                                                                       |
| <i>A</i>     | input matrix                                                          |
| <i>lda</i>   | leading dimension of A                                                |
| <i>P</i>     | LAPACK permutation                                                    |
| <i>R</i>     | first values of P                                                     |

The non pivot elements, are located in montonically increasing order.



**11.21.3.7 fgetrs() [1/4]**

```
template<class Field >
void fgetrs (
    const Field & F,
    const FFLAS::FFLAS_SIDE Side,
    const size_t M,
    const size_t N,
    const size_t R,
    typename Field::Element_ptr A,
    const size_t lda,
    const size_t * P,
    const size_t * Q,
    typename Field::Element_ptr B,
    const size_t ldb,
    int * info)
```

Solve the system  $AX = B$  or  $XA = B$ .

Solving using the PLUQ decomposition of A already computed inplace with PLUQ (FFLAS::FflasNonUnit). Version for A square. If A is rank deficient, a solution is returned if the system is consistent, Otherwise an info is 1

**Parameters**

|             |                                                                                    |
|-------------|------------------------------------------------------------------------------------|
| <i>F</i>    | base field                                                                         |
| <i>Side</i> | Determine wheter the resolution is left (FflasLeft) or right (FflasRight) looking. |
| <i>M</i>    | row dimension of B                                                                 |
| <i>N</i>    | col dimension of B                                                                 |
| <i>R</i>    | rank of A                                                                          |
| <i>A</i>    | input matrix                                                                       |
| <i>lda</i>  | leading dimension of A                                                             |
| <i>P</i>    | row permutation of the PLUQ decomposition of A                                     |
| <i>Q</i>    | column permutation of the PLUQ decomposition of A                                  |
| <i>B</i>    | Right/Left hand side matrix. Initially stores B, finally stores the solution X.    |
| <i>ldb</i>  | leading dimension of B                                                             |
| <i>info</i> | Success of the computation: 0 if successfull, >0 if system is inconsistent         |

**11.21.3.8 fgetrs() [2/4]**

```
template<class Field >
Field::Element_ptr fgetrs (
    const Field & F,
    const FFLAS::FFLAS_SIDE Side,
    const size_t M,
    const size_t N,
    const size_t NRHS,
    const size_t R,
    typename Field::Element_ptr A,
    const size_t lda,
    const size_t * P,
    const size_t * Q,
    typename Field::Element_ptr X,
    const size_t ldx,
```

```

typename Field::ConstElement_ptr B,
const size_t ldb,
int * info)

```

Solve the system  $A X = B$  or  $X A = B$ .

Solving using the PLUQ decomposition of A already computed inplace with PLUQ(FFLAS::FflasNonUnit). Version for A rectangular. If A is rank deficient, a solution is returned if the system is consistent, Otherwise an info is 1

#### Parameters

|             |                                                                                                             |
|-------------|-------------------------------------------------------------------------------------------------------------|
| <i>F</i>    | base field                                                                                                  |
| <i>Side</i> | Determine wheter the resolution is left (FflasLeft) or right (FflasRight) looking.                          |
| <i>M</i>    | row dimension of A                                                                                          |
| <i>N</i>    | col dimension of A                                                                                          |
| <i>NRHS</i> | number of columns (if Side = FFLAS::FflasLeft) or row (if Side = FFLAS::FflasRight) of the matrices X and B |
| <i>R</i>    | rank of A                                                                                                   |
| <i>A</i>    | input matrix                                                                                                |
| <i>lda</i>  | leading dimension of A                                                                                      |
| <i>P</i>    | row permutation of the PLUQ decomposition of A                                                              |
| <i>Q</i>    | column permutation of the PLUQ decomposition of A                                                           |
| <i>X</i>    | solution matrix                                                                                             |
| <i>ldx</i>  | leading dimension of X                                                                                      |
| <i>B</i>    | Right/Left hand side matrix.                                                                                |
| <i>ldb</i>  | leading dimension of B                                                                                      |
| <i>info</i> | Succes of the computation: 0 if successfull, >0 if system is inconsistent                                   |

#### 11.21.3.9 fgesv() [1/4]

```

template<class Field >
size_t fgesv (
    const Field & F,
    const FFLAS::FFLAS_SIDE Side,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr B,
    const size_t ldb,
    int * info)

```

Square system solver.

#### Parameters

|             |                                                                                   |
|-------------|-----------------------------------------------------------------------------------|
| <i>F</i>    | The computation domain                                                            |
| <i>Side</i> | Determine wheter the resolution is left (FflasLeft) or right (FflasRight) looking |
| <i>M</i>    | row dimension of B                                                                |
| <i>N</i>    | col dimension of B                                                                |
| <i>A</i>    | input matrix                                                                      |
| <i>lda</i>  | leading dimension of A                                                            |

|             |                                                                                     |
|-------------|-------------------------------------------------------------------------------------|
| <i>B</i>    | Right/Left hand side matrix. Initially contains B, finally contains the solution X. |
| <i>ldb</i>  | leading dimension of B                                                              |
| <i>info</i> | Success of the computation: 0 if successfull, >0 if system is inconsistent          |

**Returns**

the rank of the system

Solve the system  $A X = B$  or  $X A = B$ . Version for A square. If A is rank deficient, a solution is returned if the system is consistent, Otherwise an info is 1

**11.21.3.10 fgesv() [2/4]**

```
template<class Field >
size_t fgesv (
    const Field & F,
    const FFLAS::FFLAS_SIDE Side,
    const size_t M,
    const size_t N,
    const size_t NRHS,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr X,
    const size_t ldx,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    int * info)
```

Rectangular system solver.

**Parameters**

|             |                                                                                                             |
|-------------|-------------------------------------------------------------------------------------------------------------|
| <i>F</i>    | The computation domain                                                                                      |
| <i>Side</i> | Determine wheter the resolution is left (FflasLeft) or right (FflasRight) looking                           |
| <i>M</i>    | row dimension of A                                                                                          |
| <i>N</i>    | col dimension of A                                                                                          |
| <i>NRHS</i> | number of columns (if Side = FFLAS::FflasLeft) or row (if Side = FFLAS::FflasRight) of the matrices X and B |
| <i>A</i>    | input matrix                                                                                                |
| <i>lda</i>  | leading dimension of A                                                                                      |
| <i>B</i>    | Right/Left hand side matrix. Initially contains B, finally contains the solution X.                         |
| <i>ldb</i>  | leading dimension of B                                                                                      |
| <i>X</i>    |                                                                                                             |
| <i>ldx</i>  |                                                                                                             |
| <i>info</i> | Success of the computation: 0 if successfull, >0 if system is inconsistent                                  |

**Returns**

the rank of the system

Solve the system  $A X = B$  or  $X A = B$ . Version for A square. If A is rank deficient, a solution is returned if the system is consistent, Otherwise an info is 1

**11.21.3.11 ftrtri()** [1/2]

```
template<class Field >
void ftrtri (
    const Field & F,
    const FFLAS::FFLAS_UPLO Uplo,
    const FFLAS::FFLAS_DIAG Diag,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    const size_t threshold = __FFLASFFPACK_FTRTRI_THRESHOLD)
```

Compute the inverse of a triangular matrix.

**Parameters**

|             |                                                        |
|-------------|--------------------------------------------------------|
| <i>F</i>    | base field                                             |
| <i>Uplo</i> | whether the matrix is upper or lower triangular        |
| <i>Diag</i> | whether the matrix is unit diagonal (FflasUnit/NoUnit) |
| <i>N</i>    | input matrix order                                     |
| <i>A</i>    | the input matrix                                       |
| <i>lda</i>  | leading dimension of A                                 |

**11.21.3.12 trinv\_left()** [1/2]

```
template<class Field >
void trinv_left (
    const Field & F,
    const size_t N,
    typename Field::ConstElement_ptr L,
    const size_t ldl,
    typename Field::Element_ptr X,
    const size_t ldx)
```

**11.21.3.13 ftrtrm()** [1/2]

```
template<class Field >
void ftrtrm (
    const Field & F,
    const FFLAS::FFLAS_SIDE side,
    const FFLAS::FFLAS_DIAG diag,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda)
```

Compute the product of two triangular matrices of opposite shape.

Product UL or LU of the upper, resp lower triangular matrices U and L stored one above the other in the square matrix A.

## Parameters

|             |                                                                      |
|-------------|----------------------------------------------------------------------|
| <i>F</i>    | base field                                                           |
| <i>Side</i> | set to FflasLeft to compute the product UL, FflasRight to compute LU |
| <i>diag</i> | whether the matrix U is unit diagonal (FflasUnit/NoUnit)             |
| <i>N</i>    | input matrix order                                                   |
| <i>A</i>    | the input matrix                                                     |
| <i>lda</i>  | leading dimension of A                                               |

## 11.21.3.14 ftrstr()

```
template<class Field >
void ftrstr (
    const Field & F,
    const FFLAS::FFLAS_SIDE side,
    const FFLAS::FFLAS_UPLO Uplo,
    const FFLAS::FFLAS_DIAG diagA,
    const FFLAS::FFLAS_DIAG diagB,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::Element_ptr B,
    const size_t ldb,
    const size_t threshold = __FFLASFFPACK_FTRSTR_THRESHOLD) [inline]
```

Solve a triangular system with a triangular right hand side of the same shape.

## Parameters

|              |                                                                                                                       |
|--------------|-----------------------------------------------------------------------------------------------------------------------|
| <i>F</i>     | base field                                                                                                            |
| <i>Side</i>  | set to FflasLeft to compute $U1^{-1} * U2$ or $L1^{-1} * L2$ , FflasRight to compute $U1 * U2^{-1}$ or $L1 * L2^{-1}$ |
| <i>Uplo</i>  | whether the matrix A is upper or lower triangular                                                                     |
| <i>diag1</i> | whether the matrix U1 or L2 is unit diagonal (FflasUnit/NoUnit)                                                       |
| <i>diag2</i> | whether the matrix U2 or L2 is unit diagonal (FflasUnit/NoUnit)                                                       |
| <i>N</i>     | order of the input matrices                                                                                           |
| <i>A</i>     | the input matrix to be inverted (U1 or L1)                                                                            |
| <i>lda</i>   | leading dimension of A                                                                                                |
| <i>B</i>     | the input right hand side (U2 or L2)                                                                                  |
| <i>ldb</i>   | leading dimension of B                                                                                                |

## 11.21.3.15 ftrssyr2k()

```
template<class Field >
void ftrssyr2k (
    const Field & F,
    const FFLAS::FFLAS_UPLO Uplo,
    const FFLAS::FFLAS_DIAG diagA,
    const size_t N,
    typename Field::ConstElement_ptr A,
```

```

const size_t lda,
typename Field::Element_ptr B,
const size_t ldb,
const size_t threshold = __FFLASFFPACK_FTRSSYR2K_THRESHOLD) [inline]

```

Solve a triangular system in a symmetric sum: find B upper/lower triangular such that  $A^T B + B^T A = C$  where C is symmetric.

C is overwritten by B.

#### Parameters

|         |              |                                                                                                                       |
|---------|--------------|-----------------------------------------------------------------------------------------------------------------------|
|         | <i>F</i>     | base field                                                                                                            |
|         | <i>Side</i>  | set to FflasLeft to compute $U1^{-1} * U2$ or $L1^{-1} * L2$ , FflasRight to compute $U1 * U2^{-1}$ or $L1 * L2^{-1}$ |
|         | <i>Uplo</i>  | whether the matrix A is upper or lower triangular                                                                     |
|         | <i>diagA</i> | whether the matrix A is unit diagonal (FflasUnit/NoUnit)                                                              |
|         | <i>N</i>     | order of the input matrices                                                                                           |
|         | <i>A</i>     | the input matrix                                                                                                      |
|         | <i>lda</i>   | leading dimension of A                                                                                                |
| in, out | <i>B</i>     | the input right hand side where the output is written                                                                 |
|         | <i>ldb</i>   | leading dimension of B                                                                                                |

#### 11.21.3.16 fsytrf() [1/3]

```

template<class Field >
bool fsytrf (
    const Field & F,
    const FFLAS::FFLAS_UPLO UpLo,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    const size_t threshold = __FFLASFFPACK_FSYTRF_THRESHOLD)

```

Triangular factorization of symmetric matrices.

#### Parameters

|         |             |                                                                                          |
|---------|-------------|------------------------------------------------------------------------------------------|
|         | <i>F</i>    | The computation domain                                                                   |
|         | <i>UpLo</i> | Determine wheter to store the upper (FflasUpper) or lower (FflasLower) triangular factor |
|         | <i>N</i>    | order of the matrix A                                                                    |
| in, out | <i>A</i>    | input matrix                                                                             |
|         | <i>lda</i>  | leading dimension of A                                                                   |

#### Returns

false if the A does not have generic rank profile, making the computation fail.

Compute the a triangular factorization of the matrix A:  $A = L \times D \times L^T$  if UpLo = FflasLower or  $A = U^T \times D \times U$  otherwise. D is a diagonal matrix. The matrices L and U are unit diagonal lower (resp. upper) triangular and overwrite the input matrix A. The matrix D is stored on the diagonal of A, as the diagonal of L or U is known to be all ones. If A does not have generic rank profile, the LDLT or UTDU factorizations is not defined, and the algorithm returns false.

**11.21.3.17 fsytrf() [2/3]**

```
template<class Field >
bool fsytrf (
    const Field & F,
    const FFLAS::FFLAS_UPLO UpLo,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    const FFLAS::ParSeqHelper::Sequential seq,
    const size_t threshold = __FFLASFFPACK_FSYTRF_THRESHOLD) [inline]
```

**11.21.3.18 fsytrf() [3/3]**

```
template<class Field , class Cut , class Param >
bool fsytrf (
    const Field & F,
    const FFLAS::FFLAS_UPLO UpLo,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    const FFLAS::ParSeqHelper::Parallel< Cut, Param > par,
    const size_t threshold = __FFLASFFPACK_FSYTRF_THRESHOLD) [inline]
```

**11.21.3.19 fsytrf\_nonunit() [1/3]**

```
template<class Field >
bool fsytrf_nonunit (
    const Field & F,
    const FFLAS::FFLAS_UPLO UpLo,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr D,
    const size_t incD,
    const size_t threshold = __FFLASFFPACK_FSYTRF_THRESHOLD)
```

Triangular factorization of symmetric matrices.

**Parameters**

|         |        |                                                                                          |
|---------|--------|------------------------------------------------------------------------------------------|
|         | $F$    | The computation domain                                                                   |
|         | $UpLo$ | Determine wheter to store the upper (FflasUpper) or lower (FflasLower) triangular factor |
|         | $N$    | order of the matrix A                                                                    |
| in, out | $A$    | input matrix                                                                             |
| in, out | $D$    |                                                                                          |
|         | $lda$  | leading dimension of A                                                                   |

**Returns**

false if the A does not have generic rank profile, making the computation fail.

Compute the a triangular factorization of the matrix  $A$ :  $A = L \times D_{inv} \times L^T$  if  $UpLo = FflasLower$  or  $A = U^T \times D \times U$  otherwise.  $D$  is a diagonal matrix. The matrices  $L$  and  $U$  are lower (resp. upper) triangular and overwrite the input matrix  $A$ . The matrix  $D$  need to be stored separately, as the diagonal of  $L$  or  $U$  are not unit. If  $A$  does not have generic rank profile, the LDLT or UTDU factorizations is not defined, and the algorithm returns false.

**11.21.3.20 PLUQ()** [1/6]

```
template<class Field >
size_t PLUQ (
    const Field & F,
    const FFLAS::FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t * Q) [inline]
```

Compute a PLUQ factorization of the given matrix.

Return its rank. The permutations P and Q are represented using LAPACK's convention.

**Parameters**

|             |                                                                        |
|-------------|------------------------------------------------------------------------|
| <i>F</i>    | base field                                                             |
| <i>Diag</i> | whether U should have a unit diagonal (FflasUnit) or not (FflasNoUnit) |
| <i>M</i>    | matrix row dimension                                                   |
| <i>N</i>    | matrix column dimension                                                |
| <i>A</i>    | input matrix                                                           |
| <i>lda</i>  | leading dimension of A                                                 |
| <i>P</i>    | the row permutation                                                    |
| <i>Q</i>    | the column permutation                                                 |

**Returns**

the rank of A

**Bibliography** • Dumas J-G., Pernet C., and Sultan Z. *Simultaneous computation of the row and column rank profiles*, ISSAC'13, 2013

**11.21.3.21 pPLUQ()**

```
template<class Field >
size_t pPLUQ (
    const Field & F,
    const FFLAS::FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t * Q) [inline]
```



**11.21.3.22 PLUQ()** [2/6]

```
template<class Field >
size_t PLUQ (
    const Field & F,
    const FFLAS::FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t * Q,
    const FFLAS::ParSeqHelper::Sequential & PSHelper,
    size_t BCThreshold = __FFLASFFPACK_PLUQ_THRESHOLD) [inline]
```

**11.21.3.23 PLUQ()** [3/6]

```
template<class Field , class Cut , class Param >
size_t PLUQ (
    const Field & F,
    const FFLAS::FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t * Q,
    const FFLAS::ParSeqHelper::Parallel< Cut, Param > & PSHelper)
```

**11.21.3.24 LUdivine()** [1/4]

```
template<class Field >
size_t LUdivine (
    const Field & F,
    const FFLAS::FFLAS_DIAG Diag,
    const FFLAS::FFLAS_TRANSPOSE trans,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const FFPACK_LU_TAG LuTag = FfpackSlabRecursive,
    const size_t cutoff = __FFLASFFPACK_LUDIVINE_THRESHOLD)
```

Compute the CUP or PLE factorization of the given matrix.

Using a block algorithm and return its rank. The permutations P and Q are represented using LAPACK's convention.

**Parameters**

|             |                                                                                                                             |
|-------------|-----------------------------------------------------------------------------------------------------------------------------|
| <i>F</i>    | base field                                                                                                                  |
| <i>Diag</i> | whether the transformation matrix (U of the CUP, L of the PLE) should have a unit diagonal (FflasUnit) or not (FflasNoUnit) |

## Parameters

|               |                                                                                                 |
|---------------|-------------------------------------------------------------------------------------------------|
| <i>trans</i>  | whether to compute the CUP decomposition (FflasNoTrans) or the PLE decomposition (FflasTrans)   |
| <i>M</i>      | matrix row dimension                                                                            |
| <i>N</i>      | matrix column dimension                                                                         |
| <i>A</i>      | input matrix                                                                                    |
| <i>lda</i>    | leading dimension of A                                                                          |
| <i>P</i>      | the factor of CUP or PLE                                                                        |
| <i>Q</i>      | a permutation indicating the pivot position in the echelon form C or E in its first r positions |
| <i>LuTag</i>  | flag for setting the earling termination if the matrix is singular                              |
| <i>cutoff</i> | threshold to basecase                                                                           |

## Returns

the rank of A

## Bibliography

- Jeannerod C-P, Pernet, C. and Storjohann, A. *Rank-profile revealing Gaussian elimination and the CUP matrix decomposition*, J. of Symbolic Comp., 2013
- Pernet C, Brassel M *LUdivine, une divine factorisation LU*, 2002

## 11.21.3.25 ColumnEchelonForm() [1/3]

```
template<class Field >
size_t ColumnEchelonForm (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    bool transform = false,
    const FFPACK_LU_TAG LuTag = FfpackSlabRecursive) [inline]
```

Compute the Column Echelon form of the input matrix in-place.

If  $LuTag == FfpackTileRecursive$ , then after the computation  $A = [M \setminus V]$  such that  $AU = C$  is a column echelon decomposition of A, with  $U = P^T [V]$  and  $C = M + Q [I_r] [0 \text{ } I_{n-r}] [0]$  If  $LuTag == FfpackTileRecursive$  then  $A = [N \setminus V]$  such that the same holds with  $M = Q N$

$Qt = Q^T$  If  $transform=false$ , the matrix V is not computed. See also test-colechelon for an example of use

## Parameters

|    |                  |                                                                                       |
|----|------------------|---------------------------------------------------------------------------------------|
|    | <i>F</i>         | base field                                                                            |
|    | <i>M</i>         | number of rows                                                                        |
|    | <i>N</i>         | number of columns                                                                     |
| in | <i>A</i>         | input matrix                                                                          |
|    | <i>lda</i>       | leading dimension of A                                                                |
|    | <i>P</i>         | the column permutation                                                                |
|    | <i>Qt</i>        | the row position of the pivots in the echelon form                                    |
|    | <i>transform</i> | decides whether V is computed                                                         |
|    | <i>LuTag</i>     | chooses the elimination algorithm. SlabRecursive for LUdivine, TileRecursive for PLUQ |

## 11.21.3.26 pColumnEchelonForm()

```
template<class Field >
size_t pColumnEchelonForm (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    bool transform = false,
    size_t numthreads = 0,
    const FFPACK_LU_TAG LuTag = FfpackTileRecursive) [inline]
```

## 11.21.3.27 ColumnEchelonForm() [2/3]

```
template<class Field , class PSHelper >
size_t ColumnEchelonForm (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const FFPACK_LU_TAG LuTag,
    const PSHelper & psH) [inline]
```

## 11.21.3.28 RowEchelonForm() [1/3]

```
template<class Field >
size_t RowEchelonForm (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform = false,
    const FFPACK_LU_TAG LuTag = FfpackSlabRecursive) [inline]
```

Compute the Row Echelon form of the input matrix in-place.

If  $\text{LuTag} == \text{FfpackTileRecursive}$ , then after the computation  $A = [L \setminus M]$  such that  $XA = R$  is a row echelon decomposition of  $A$ , with  $X = [L \ 0]P$  and  $R = M + [I \ 0]Q^T [In-r]$  If  $\text{LuTag} == \text{FfpackTileRecursive}$  then  $A = [L \setminus N]$  such that the same holds with  $M = NQ^TQt = Q^T$  If  $\text{transform} = \text{false}$ , the matrix  $L$  is not computed. See also `test-rowechelon` for an example of use

## Parameters

|  |     |            |
|--|-----|------------|
|  | $F$ | base field |
|--|-----|------------|

## Parameters

|    |                  |                                                                                       |
|----|------------------|---------------------------------------------------------------------------------------|
|    | $M$              | number of rows                                                                        |
|    | $N$              | number of columns                                                                     |
| in | $A$              | the input matrix                                                                      |
|    | $lda$            | leading dimension of $A$                                                              |
|    | $P$              | the row permutation                                                                   |
|    | $Qt$             | the column position of the pivots in the echelon form                                 |
|    | <i>transform</i> | decides whether $L$ is computed                                                       |
|    | <i>LuTag</i>     | chooses the elimination algorithm. SlabRecursive for LUdivine, TileRecursive for PLUQ |

## 11.21.3.29 pRowEchelonForm()

```
template<class Field >
size_t pRowEchelonForm (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform = false,
    size_t numthreads = 0,
    const FFPACK_LU_TAG LuTag = FfpackTileRecursive) [inline]
```

## 11.21.3.30 RowEchelonForm() [2/3]

```
template<class Field , class PSHelper >
size_t RowEchelonForm (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const FFPACK_LU_TAG LuTag,
    const PSHelper & psH) [inline]
```

## 11.21.3.31 ReducedColumnEchelonForm() [1/3]

```
template<class Field >
size_t ReducedColumnEchelonForm (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
```

```

size_t * P,
size_t * Qt,
const bool transform = false,
const FFPACK_LU_TAG LuTag = FfpackSlabRecursive) [inline]

```

Compute the Reduced Column Echelon form of the input matrix in-place.

After the computation  $A = [V]$  such that  $AX = R$  is a reduced col echelon  $[M \ 0]$  decomposition of  $A$ , where  $X = P^T [V]$  and  $R = Q [I_r \ 0 \ I_{n-r}] [M \ 0] Qt = Q^T$  If transform=false, the matrix  $X$  is not computed and the matrix  $A = R$

#### Parameters

|    |                  |                                                                                       |
|----|------------------|---------------------------------------------------------------------------------------|
|    | <i>F</i>         | base field                                                                            |
|    | <i>M</i>         | number of rows                                                                        |
|    | <i>N</i>         | number of columns                                                                     |
| in | <i>A</i>         | input matrix                                                                          |
|    | <i>lda</i>       | leading dimension of A                                                                |
|    | <i>P</i>         | the column permutation                                                                |
|    | <i>Qt</i>        | the row position of the pivots in the echelon form                                    |
|    | <i>transform</i> | decides whether X is computed                                                         |
|    | <i>LuTag</i>     | chooses the elimination algorithm. SlabRecursive for LUdivine, TileRecursive for PLUQ |

#### 11.21.3.32 pReducedColumnEchelonForm()

```

template<class Field >
size_t pReducedColumnEchelonForm (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform = false,
    size_t numthreads = 0,
    const FFPACK_LU_TAG LuTag = FfpackTileRecursive) [inline]

```

#### 11.21.3.33 ReducedColumnEchelonForm() [2/3]

```

template<class Field , class PSHelper >
size_t ReducedColumnEchelonForm (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const FFPACK_LU_TAG LuTag,
    const PSHelper & psH) [inline]

```

**11.21.3.34 ReducedRowEchelonForm()** [1/3]

```
template<class Field >
size_t ReducedRowEchelonForm (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform = false,
    const FFPACK_LU_TAG LuTag = FfpackSlabRecursive) [inline]
```

Compute the Reduced Row Echelon form of the input matrix in-place.

After the computation  $A = [V1 \ M]$  such that  $X \ A = R$  is a reduced row echelon  $[V2 \ 0]$  decomposition of  $A$ , where  $X = [V1 \ 0] \ P$  and  $R = [I_r \ M] \ Q^T$   $[V2 \ In-r] \ [0] \ Qt = Q^T$  If  $transform=false$ , the matrix  $X$  is not computed and the matrix  $A = R$

**Parameters**

|    |             |                                                                                       |
|----|-------------|---------------------------------------------------------------------------------------|
|    | $F$         | base field                                                                            |
|    | $M$         | number of rows                                                                        |
|    | $N$         | number of columns                                                                     |
| in | $A$         | input matrix                                                                          |
|    | $lda$       | leading dimension of $A$                                                              |
|    | $P$         | the row permutation                                                                   |
|    | $Qt$        | the column position of the pivots in the echelon form                                 |
|    | $transform$ | decides whether $X$ is computed                                                       |
|    | $LuTag$     | chooses the elimination algorithm. SlabRecursive for LUdivine, TileRecursive for PLUQ |

**11.21.3.35 pReducedRowEchelonForm()**

```
template<class Field >
size_t pReducedRowEchelonForm (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform = false,
    size_t numthreads = 0,
    const FFPACK_LU_TAG LuTag = FfpackTileRecursive) [inline]
```

**11.21.3.36 ReducedRowEchelonForm()** [2/3]

```
template<class Field , class PSHelper >
size_t ReducedRowEchelonForm (
    const Field & F,
```

```

const size_t M,
const size_t N,
typename Field::Element_ptr A,
const size_t lda,
size_t * P,
size_t * Qt,
const bool transform,
const FFPACK_LU_TAG LuTag,
const PSHelper & psH) [inline]

```

### 11.21.3.37 Invert() [1/4]

```

template<class Field >
Field::Element_ptr Invert (
    const Field & F,
    const size_t M,
    typename Field::Element_ptr A,
    const size_t lda,
    int & nullity)

```

Invert the given matrix in place or computes its nullity if it is singular.

An inplace  $2n^3$  algorithm is used.

#### Parameters

|         |           |                               |
|---------|-----------|-------------------------------|
|         | $F$       | The computation domain        |
|         | $M$       | order of the matrix           |
| in, out | $A$       | input matrix ( $M \times M$ ) |
|         | $lda$     | leading dimension of A        |
|         | $nullity$ | dimension of the kernel of A  |

#### Returns

pointer to  $A$  and  $A \leftarrow A^{-1}$

### 11.21.3.38 Invert() [2/4]

```

template<class Field >
Field::Element_ptr Invert (
    const Field & F,
    const size_t M,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::Element_ptr X,
    const size_t ldx,
    int & nullity)

```

Invert the given matrix or computes its nullity if it is singular.

#### Precondition

$X$  is preallocated and should be large enough to store the  $m \times m$  matrix  $A$ .

## Parameters

|     |           |                                                                                                    |
|-----|-----------|----------------------------------------------------------------------------------------------------|
|     | $F$       | The computation domain                                                                             |
|     | $M$       | order of the matrix                                                                                |
| in  | $A$       | input matrix ( $M \times M$ )                                                                      |
|     | $lda$     | leading dimension of A                                                                             |
| out | $X$       | this is the inverse of A if A is invertible (non NULL and nullity = 0). It is untouched otherwise. |
|     | $ldx$     | leading dimension of X                                                                             |
|     | $nullity$ | dimension of the kernel of A                                                                       |

## Returns

pointer to  $X = A^{-1}$

## 11.21.3.39 Invert2() [1/2]

```
template<class Field >
Field::Element_ptr Invert2 (
    const Field & F,
    const size_t M,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr X,
    const size_t ldx,
    int & nullity)
```

Invert the given matrix or computes its nullity if it is singular.

An  $2n^3f$  algorithm is used. This routine can be % faster than [FFPACK::Invert](#) but is not totally inplace.

## Precondition

X is preallocated and should be large enough to store the  $m \times m$  matrix A.

## Warning

A is overwritten here !

**Bug** not tested.

## Parameters

|         |           |                                                                                                            |
|---------|-----------|------------------------------------------------------------------------------------------------------------|
|         | $F$       | the computation domain                                                                                     |
|         | $M$       | order of the matrix                                                                                        |
| in, out | $A$       | input matrix ( $M \times M$ ). On output, A is modified and represents a "psychological" factorisation LU. |
|         | $lda$     | leading dimension of A                                                                                     |
| out     | $X$       | this is the inverse of A if A is invertible (non NULL and nullity = 0). It is untouched otherwise.         |
|         | $ldx$     | leading dimension of X                                                                                     |
|         | $nullity$ | dimension of the kernel of A                                                                               |



## Returns

pointer to  $X = A^{-1}$

**Todo** this init is not all necessary (done after ftrtri)

**Todo** this init is not all necessary (done after ftrtri)

## 11.21.3.40 CharPoly() [1/8]

```
template<class PolRing >
std::list< typename PolRing::Element > & CharPoly (
    const PolRing & R,
    std::list< typename PolRing::Element > & charp,
    const size_t N,
    typename PolRing::Domain_t::Element_ptr A,
    const size_t lda,
    typename PolRing::Domain_t::RandIter & G,
    const FFPACK_CHARPOLY_TAG CharpTag = FfpackAuto,
    const size_t degree = __FFLASFFPACK_ARITHPROG_THRESHOLD) [inline]
```

Compute the characteristic polynomial of the matrix A.

## Parameters

|     |                 |                                                                                       |
|-----|-----------------|---------------------------------------------------------------------------------------|
|     | <i>R</i>        | the polynomial ring of charp (contains the base field)                                |
| out | <i>charp</i>    | the characteristic polynomial of as a list of factors                                 |
|     | <i>N</i>        | order of the matrix A                                                                 |
| in  | <i>A</i>        | the input matrix ( $N \times N$ ) (could be overwritten in some algorithmic variants) |
|     | <i>lda</i>      | leading dimension of A                                                                |
|     | <i>CharpTag</i> | the algorithmic variant                                                               |
|     | <i>G</i>        | a random iterator (required for the randomized variants LUKrylov and ArithProg)       |

## 11.21.3.41 CharPoly() [2/8]

```
template<class PolRing >
PolRing::Element & CharPoly (
    const PolRing & R,
    typename PolRing::Element & charp,
    const size_t N,
    typename PolRing::Domain_t::Element_ptr A,
    const size_t lda,
    typename PolRing::Domain_t::RandIter & G,
    const FFPACK_CHARPOLY_TAG CharpTag = FfpackAuto,
    const size_t degree = __FFLASFFPACK_ARITHPROG_THRESHOLD) [inline]
```

Compute the characteristic polynomial of the matrix A.

## Parameters

|  |          |                                                        |
|--|----------|--------------------------------------------------------|
|  | <i>R</i> | the polynomial ring of charp (contains the base field) |
|--|----------|--------------------------------------------------------|

|     |                 |                                                                                       |
|-----|-----------------|---------------------------------------------------------------------------------------|
| out | <i>charp</i>    | the characteristic polynomial of <i>A</i> as a single polynomial                      |
|     | <i>N</i>        | order of the matrix <i>A</i>                                                          |
| in  | <i>A</i>        | the input matrix ( $N \times N$ ) (could be overwritten in some algorithmic variants) |
|     | <i>lda</i>      | leading dimension of <i>A</i>                                                         |
|     | <i>CharpTag</i> | the algorithmic variant                                                               |
|     | <i>G</i>        | a random iterator (required for the randomized variants LUKrylov and ArithProg)       |

#### 11.21.3.42 CharPoly() [3/8]

```
template<class PolRing >
PolRing::Element & CharPoly (
    const PolRing & R,
    typename PolRing::Element & charp,
    const size_t N,
    typename PolRing::Domain_t::Element_ptr A,
    const size_t lda,
    const FFPACK_CHARPOLY_TAG CharpTag = FfpackAuto,
    const size_t degree = __FFLASFFPACK_ARITHPROG_THRESHOLD) [inline]
```

Compute the characteristic polynomial of the matrix *A*.

##### Parameters

|     |                 |                                                                                       |
|-----|-----------------|---------------------------------------------------------------------------------------|
|     | <i>R</i>        | the polynomial ring of <i>charp</i> (contains the base field)                         |
| out | <i>charp</i>    | the characteristic polynomial of <i>A</i> as a single polynomial                      |
|     | <i>N</i>        | order of the matrix <i>A</i>                                                          |
| in  | <i>A</i>        | the input matrix ( $N \times N$ ) (could be overwritten in some algorithmic variants) |
|     | <i>lda</i>      | leading dimension of <i>A</i>                                                         |
|     | <i>CharpTag</i> | the algorithmic variant                                                               |

#### 11.21.3.43 MinPoly() [1/4]

```
template<class Field , class Polynomial >
Polynomial & MinPoly (
    const Field & F,
    Polynomial & minP,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t lda) [inline]
```

Compute the minimal polynomial of the matrix *A*.

The algorithm is randomized probabilistic, and computes the minimal polynomial of the Krylov iterates of a random vector: (v, Av, ..., A<sup>k</sup>v)

##### Parameters

|     |             |                                    |
|-----|-------------|------------------------------------|
|     | <i>F</i>    | the base field                     |
| out | <i>minP</i> | the minimal polynomial of <i>A</i> |
|     | <i>N</i>    | order of the matrix <i>A</i>       |
| in  | <i>A</i>    | the input matrix ( $N \times N$ )  |
|     | <i>lda</i>  | leading dimension of <i>A</i>      |

**11.21.3.44 MinPoly()** [2/4]

```
template<class Field , class Polynomial , class RandIter >
Polynomial & MinPoly (
    const Field & F,
    Polynomial & minP,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    RandIter & G) [inline]
```

Compute the minimal polynomial of the matrix A.

The algorithm is randomized probabilistic, and computes the minimal polynomial of the Krylov iterates of a random vector:  $(v, Av, \dots, A^N v)$

**Parameters**

|     |        |                                   |
|-----|--------|-----------------------------------|
|     | $F$    | the base field                    |
| out | $minP$ | the minimal polynomial of A       |
|     | $N$    | order of the matrix A             |
| in  | $A$    | the input matrix ( $N \times N$ ) |
|     | $lda$  | leading dimension of A            |
|     | $G$    | a random iterator                 |

**11.21.3.45 MatVecMinPoly()** [1/2]

```
template<class Field , class Polynomial >
Polynomial & MatVecMinPoly (
    const Field & F,
    Polynomial & minP,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr v,
    const size_t incv) [inline]
```

Compute the minimal polynomial of the matrix A and a vector v, namely the first linear dependency relation in the Krylov basis  $(v, Av, \dots, A^N v)$ .

**Parameters**

|     |        |                                                                                 |
|-----|--------|---------------------------------------------------------------------------------|
|     | $F$    | the base field                                                                  |
| out | $minP$ | the minimal polynomial of A and v                                               |
|     | $N$    | order of the matrix A                                                           |
| in  | $A$    | the input matrix ( $N \times N$ )                                               |
|     | $lda$  | leading dimension of A                                                          |
|     | $K$    | an $N \times (N + 1)$ matrix containing the vector v on its first row           |
|     | $ldk$  | leading dimension of K                                                          |
|     | $P$    | [out] (optional) the permutation used in the elimination of the Krylov matrix K |

**11.21.3.46 Rank()** [1/3]

```
template<class Field >
size_t Rank (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda)
```

Computes the rank of the given matrix using a PLUQ factorization.

The input matrix is modified.

**Parameters**

|    |            |                                                                               |
|----|------------|-------------------------------------------------------------------------------|
|    | <i>F</i>   | base field                                                                    |
|    | <i>M</i>   | row dimension of the matrix                                                   |
|    | <i>N</i>   | column dimension of the matrix                                                |
| in | <i>A</i>   | input matrix                                                                  |
|    | <i>lda</i> | leading dimension of A                                                        |
|    | <i>psH</i> | (optional) a ParSeqHelper to choose between sequential and parallel execution |

**11.21.3.47 pRank()**

```
template<class Field >
size_t pRank (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t numthreads = 0)
```

**11.21.3.48 Rank()** [2/3]

```
template<class Field , class PSHelper >
size_t Rank (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    const PSHelper & psH)
```

**11.21.3.49 IsSingular()** [1/2]

```
template<class Field >
bool IsSingular (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda)
```

Returns true if the given matrix is singular.

The method is a block elimination with early termination

using LQUP factorization with early termination. If  $M \neq N$ , then the matrix is virtually padded with zeros to make it square and it's determinant is zero.

**Warning**

The input matrix is modified.

**Parameters**

|         |       |                                 |
|---------|-------|---------------------------------|
|         | $F$   | base field                      |
|         | $M$   | row dimension of the matrix     |
|         | $N$   | column dimension of the matrix. |
| in, out | $A$   | input matrix                    |
|         | $lda$ | leading dimension of A          |

**11.21.3.50 Det()** [1/6]

```
template<class Field >
Field::Element & Det (
    const Field & F,
    typename Field::Element & det,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P = NULL,
    size_t * Q = NULL) [inline]
```

Returns the determinant of the given square matrix.

The method is a block elimination using PLUQ factorization. The input matrix A is overwritten.

**Warning**

The input matrix is modified.

**Parameters**

|         |        |                                                                                                                                                            |
|---------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
|         | $F$    | base field                                                                                                                                                 |
| out     | $det$  | the determinant of A                                                                                                                                       |
|         | $N$    | the order of the square matrix A.                                                                                                                          |
| in, out | $A$    | input matrix                                                                                                                                               |
|         | $lda$  | leading dimension of A                                                                                                                                     |
|         | $psH$  | (optional) a ParSeqHelper to choose between sequential and parallel execution                                                                              |
|         | $P, Q$ | (optional) row and column permutations to be used by the PLUQ factorization. randomized checkers (see checker/checker_det.inl) need them for certification |

### 11.21.3.51 pDet()

```
template<class Field >
Field::Element & pDet (
    const Field & F,
    typename Field::Element & det,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t numthreads = 0,
    size_t * P = NULL,
    size_t * Q = NULL) [inline]
```

### 11.21.3.52 Det() [2/6]

```
template<class Field , class PSHelper >
Field::Element & Det (
    const Field & F,
    typename Field::Element & det,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    const PSHelper & psH,
    size_t * P = NULL,
    size_t * Q = NULL)
```

### 11.21.3.53 Solve() [1/3]

```
template<class Field >
Field::Element_ptr Solve (
    const Field & F,
    const size_t M,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr x,
    const int incx,
    typename Field::ConstElement_ptr b,
    const int incb) [inline]
```

Solves a linear system  $AX = b$  using PLUQ factorization.

@oaram F base field @oaram M matrix order

#### Parameters

|     |             |                                     |
|-----|-------------|-------------------------------------|
| in  | <i>A</i>    | input matrix                        |
|     | <i>lda</i>  | leading dimension of A              |
| out | <i>x</i>    | output solution vector              |
|     | <i>incx</i> | increment of x                      |
|     | <i>b</i>    | input right hand side of the system |
|     | <i>incb</i> | increment of b                      |

**11.21.3.54 Solve()** [2/3]

```
template<class Field , class PSHelper >
Field::Element_ptr Solve (
    const Field & F,
    const size_t M,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr x,
    const int incx,
    typename Field::ConstElement_ptr b,
    const int incb,
    PSHelper & pSH)
```

**11.21.3.55 pSolve()**

```
template<class Field >
Field::Element_ptr pSolve (
    const Field & F,
    const size_t M,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr x,
    const int incx,
    typename Field::ConstElement_ptr b,
    const int incb,
    size_t numthreads = 0) [inline]
```

**11.21.3.56 RandomNullSpaceVector()** [1/3]

```
template<class Field >
*void RandomNullSpaceVector (
    const Field & F,
    const FFLAS::FFLAS_SIDE Side,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr X,
    const size_t incX)
```

Solve  $LX = B$  or  $XL = B$  in place.

L is  $M \times M$  if Side == [FFLAS::FflasLeft](#) and  $N \times N$  if Side == [FFLAS::FflasRight](#), B is  $M \times N$ . Only the R non trivial column of L are stored in the  $M \times R$  matrix L Requirement : so that L could be expanded in-place Computes a vector of the Left/Right nullspace of the matrix A.

**Parameters**

|         |             |                                                                                  |
|---------|-------------|----------------------------------------------------------------------------------|
|         | <i>F</i>    | The computation domain                                                           |
|         | <i>Side</i> | decides whether it computes the left (FflasLeft) or right (FflasRight) nullspace |
|         | <i>M</i>    | number of rows                                                                   |
|         | <i>N</i>    | number of columns                                                                |
| in, out | <i>A</i>    | input matrix of dimension $M \times N$ , A is modified to its LU version         |
|         | <i>lda</i>  | leading dimension of A                                                           |
| out     | <i>X</i>    | output vector                                                                    |
|         | <i>incX</i> | increment of X                                                                   |

### 11.21.3.57 NullSpaceBasis() [1/2]

```
template<class Field >
size_t NullSpaceBasis (
    const Field & F,
    const FFLAS::FFLAS_SIDE Side,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr & NS,
    size_t & ldn,
    size_t & NSdim)
```

Computes a basis of the Left/Right nullspace of the matrix A.

return the dimension of the nullspace.

#### Parameters

|         |              |                                                                                  |
|---------|--------------|----------------------------------------------------------------------------------|
|         | <i>F</i>     | The computation domain                                                           |
|         | <i>Side</i>  | decides whether it computes the left (FflasLeft) or right (FflasRight) nullspace |
|         | <i>M</i>     | number of rows                                                                   |
|         | <i>N</i>     | number of columns                                                                |
| in, out | <i>A</i>     | input matrix of dimension M x N, A is modified                                   |
|         | <i>lda</i>   | leading dimension of A                                                           |
| out     | <i>NS</i>    | output matrix of dimension N x NSdim (allocated here)                            |
| out     | <i>ldn</i>   | leading dimension of NS                                                          |
| out     | <i>NSdim</i> | the dimension of the Nullspace (N-rank(A))                                       |

### 11.21.3.58 RowRankProfile() [1/3]

```
template<class Field >
size_t RowRankProfile (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t *& rkprofile,
    const FFPACK_LU_TAG LuTag = FfpackSlabRecursive) [inline]
```

Computes the row rank profile of A.

#### Parameters

|     |                  |                                                                                       |
|-----|------------------|---------------------------------------------------------------------------------------|
|     | <i>F</i>         | base field                                                                            |
|     | <i>M</i>         | number of rows                                                                        |
|     | <i>N</i>         | number of columns                                                                     |
| in  | <i>A</i>         | input matrix of dimension M x N                                                       |
|     | <i>lda</i>       | leading dimension of A                                                                |
| out | <i>rkprofile</i> | return the rank profile as an array of row indexes, of dimension r=rank(A)            |
|     | <i>LuTag</i>     | chooses the elimination algorithm. SlabRecursive for LUdivine, TileRecursive for PLUQ |

A is modified rkprofile is allocated during the computation.



## Returns

R

## 11.21.3.59 pRowRankProfile()

```
template<class Field >
size_t pRowRankProfile (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t *rkprofile,
    size_t numthreads = 0,
    const FFPACK_LU_TAG LuTag = FfpackTileRecursive) [inline]
```

## 11.21.3.60 RowRankProfile() [2/3]

```
template<class Field , class PSHelper >
size_t RowRankProfile (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t *rkprofile,
    const FFPACK_LU_TAG LuTag,
    PSHelper & psH) [inline]
```

## 11.21.3.61 ColumnRankProfile() [1/3]

```
template<class Field >
size_t ColumnRankProfile (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t *rkprofile,
    const FFPACK_LU_TAG LuTag = FfpackSlabRecursive) [inline]
```

Computes the column rank profile of A.

## Parameters

|     |             |                                                                                       |
|-----|-------------|---------------------------------------------------------------------------------------|
|     | $F$         | base field                                                                            |
|     | $M$         | number of rows                                                                        |
|     | $N$         | number of columns                                                                     |
| in  | $A$         | input matrix of dimension                                                             |
|     | $lda$       | leading dimension of A                                                                |
| out | $rkprofile$ | return the rank profile as an array of row indexes, of dimension $r=\text{rank}(A)$   |
|     | $LuTag$     | chooses the elimination algorithm. SlabRecursive for LUdivine, TileRecursive for PLUQ |

A is modified rkprofile is allocated during the computation.

## Returns

R

## 11.21.3.62 pColumnRankProfile()

```
template<class Field >
size_t pColumnRankProfile (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t *rkprofile,
    size_t numthreads = 0,
    const FFPACK_LU_TAG LuTag = FfpackTileRecursive) [inline]
```

## 11.21.3.63 ColumnRankProfile() [2/3]

```
template<class Field , class PSHelper >
size_t ColumnRankProfile (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t *rkprofile,
    const FFPACK_LU_TAG LuTag,
    PSHelper & psH) [inline]
```

## 11.21.3.64 RankProfileFromLU()

```
void RankProfileFromLU (
    const size_t * P,
    const size_t N,
    const size_t R,
    size_t * rkprofile,
    const FFPACK_LU_TAG LuTag) [inline]
```

Recovers the column/row rank profile from the permutation of an LU decomposition.

Works with both the CUP/PLE decompositions (obtained by LUdivine) or the PLUQ decomposition. Assumes that the output vector containing the rank profile is already allocated.

## Parameters

|     |             |                                                                                       |
|-----|-------------|---------------------------------------------------------------------------------------|
|     | $P$         | the permutation carrying the rank profile information                                 |
|     | $N$         | the row/col dimension for a row/column rank profile                                   |
|     | $R$         | the rank of the matrix                                                                |
| out | $rkprofile$ | return the rank profile as an array of indices                                        |
|     | $LuTag$     | chooses the elimination algorithm. SlabRecursive for LUdivine, TileRecursive for PLUQ |

**11.21.3.65 LeadingSubmatrixRankProfiles()**

```
size_t LeadingSubmatrixRankProfiles (
    const size_t M,
    const size_t N,
    const size_t R,
    const size_t LSm,
    const size_t LSn,
    const size_t * P,
    const size_t * Q,
    size_t * RRP,
    size_t * CRP) [inline]
```

Recovers the row and column rank profiles of any leading submatrix from the PLUQ decomposition.

Only works with the PLUQ decomposition Assumes that the output vectors containing the rank profiles are already allocated.

**Parameters**

|            |                                                          |
|------------|----------------------------------------------------------|
| <i>P</i>   | the permutation carrying the rank profile information    |
| <i>M</i>   | the row dimension of the initial matrix                  |
| <i>N</i>   | the column dimension of the initial matrix               |
| <i>R</i>   | the rank of the initial matrix                           |
| <i>LSm</i> | the row dimension of the leading submatrix considered    |
| <i>LSn</i> | the column dimension of the leading submatrix considered |
| <i>P</i>   | the row permutation of the PLUQ decomposition            |
| <i>Q</i>   | the column permutation of the PLUQ decomposition         |
| <i>RRP</i> | return the row rank profile of the leading submatrix     |

**Returns**

the rank of the LSm x LSn leading submatrix

A is modified

**Bibliography** • Dumas J-G., Pernet C., and Sultan Z. *Simultaneous computation of the row and column rank profiles*, ISSAC'13.

**11.21.3.66 RowRankProfileSubmatrixIndices()** [1/2]

```
template<class Field >
size_t RowRankProfileSubmatrixIndices (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * rowindices,
    size_t * colindices,
    size_t & R)
```

RowRankProfileSubmatrixIndices.

Computes the indices of the submatrix  $r \times r$  X of A whose rows correspond to the row rank profile of A.

## Parameters

|     |                   |                                    |
|-----|-------------------|------------------------------------|
|     | $F$               | base field                         |
|     | $M$               | number of rows                     |
|     | $N$               | number of columns                  |
| in  | $A$               | input matrix of dimension          |
|     | <i>rowindices</i> | array of the row indices of X in A |
|     | <i>colindices</i> | array of the col indices of X in A |
|     | <i>lda</i>        | leading dimension of A             |
| out | $R$               | list of indices                    |

rowindices and colindices are allocated during the computation. A is modified

## Returns

R

## 11.21.3.67 ColRankProfileSubmatrixIndices() [1/2]

```
template<class Field >
size_t ColRankProfileSubmatrixIndices (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t *& rowindices,
    size_t *& colindices,
    size_t & R)
```

Computes the indices of the submatrix  $r \times r$  X of A whose columns correspond to the column rank profile of A.

## Parameters

|     |                   |                                    |
|-----|-------------------|------------------------------------|
|     | $F$               | base field                         |
|     | $M$               | number of rows                     |
|     | $N$               | number of columns                  |
| in  | $A$               | input matrix of dimension          |
|     | <i>rowindices</i> | array of the row indices of X in A |
|     | <i>colindices</i> | array of the col indices of X in A |
|     | <i>lda</i>        | leading dimension of A             |
| out | $R$               | list of indices                    |

rowindices and colindices are allocated during the computation.

## Warning

A is modified

## Returns

R

**11.21.3.68 RowRankProfileSubmatrix()** [1/2]

```
template<class Field >
size_t RowRankProfileSubmatrix (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr & X,
    size_t & R)
```

Computes the  $r \times r$  submatrix  $X$  of  $A$ , by picking the row rank profile rows of  $A$ .

**Parameters**

|     |       |                                        |
|-----|-------|----------------------------------------|
|     | $F$   | base field                             |
|     | $M$   | number of rows                         |
|     | $N$   | number of columns                      |
| in  | $A$   | input matrix of dimension $M \times N$ |
|     | $lda$ | leading dimension of $A$               |
| out | $X$   | the output matrix                      |
| out | $R$   | list of indices                        |

$A$  is not modified  $X$  is allocated during the computation.

**Returns**

$R$

**11.21.3.69 ColRankProfileSubmatrix()** [1/2]

```
template<class Field >
size_t ColRankProfileSubmatrix (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr & X,
    size_t & R)
```

Compute the  $r \times r$  submatrix  $X$  of  $A$ , by picking the row rank profile rows of  $A$ .

**Parameters**

|     |       |                                        |
|-----|-------|----------------------------------------|
|     | $F$   | base field                             |
|     | $M$   | number of rows                         |
|     | $N$   | number of columns                      |
| in  | $A$   | input matrix of dimension $M \times N$ |
|     | $lda$ | leading dimension of $A$               |
| out | $X$   | the output matrix                      |
| out | $R$   | list of indices                        |

$A$  is not modified  $X$  is allocated during the computation.

## Returns

R

## 11.21.3.70 getTriangular() [1/2]

```

template<class Field >
void getTriangular (
    const Field & F,
    const FFLAS::FFLAS_UPLO UpLo,
    const FFLAS::FFLAS_DIAG diag,
    const size_t M,
    const size_t N,
    const size_t R,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::Element_ptr T,
    const size_t ldt,
    const bool OnlyNonZeroVectors = false) [inline]

```

Extracts a triangular matrix from a compact storage  $A=L\backslash U$  of rank R.

if OnlyNonZeroVectors is false, then T and A have the same dimensions Otherwise, T is R x N if UpLo = FflasUpper, else T is M x R

## Parameters

|     |                           |                                                                                       |
|-----|---------------------------|---------------------------------------------------------------------------------------|
|     | <i>F</i>                  | base field                                                                            |
|     | <i>UpLo</i>               | selects if the upper (FflasUpper) or lower (FflasLower) triangular matrix is returned |
|     | <i>diag</i>               | selects if the triangular matrix unit-diagonal (FflasUnit/NoUnit)                     |
|     | <i>M</i>                  | row dimension of T                                                                    |
|     | <i>N</i>                  | column dimension of T                                                                 |
|     | <i>R</i>                  | rank of the triangular matrix (how many rows/columns need to be copied)               |
| in  | <i>A</i>                  | input matrix                                                                          |
|     | <i>lda</i>                | leading dimension of A                                                                |
| out | <i>T</i>                  | output matrix                                                                         |
|     | <i>ldt</i>                | leading dimension of T                                                                |
|     | <i>OnlyNonZeroVectors</i> | decides whether the last zero rows/columns should be ignored                          |

**Todo** just one triangular fzero+fassign ?

**Todo** just one triangular fzero+fassign ?

**11.21.3.71 getTriangular() [2/2]**

```
template<class Field >
void getTriangular (
    const Field & F,
    const FFLAS::FFLAS_UPLO Uplo,
    const FFLAS::FFLAS_DIAG diag,
    const size_t M,
    const size_t N,
    const size_t R,
    typename Field::Element_ptr A,
    const size_t lda) [inline]
```

Cleans up a compact storage  $A=LU$  to reveal a triangular matrix of rank  $R$ .

**Parameters**

|         |             |                                                                                       |
|---------|-------------|---------------------------------------------------------------------------------------|
|         | <i>F</i>    | base field                                                                            |
|         | <i>UpLo</i> | selects if the upper (FflasUpper) or lower (FflasLower) triangular matrix is revealed |
|         | <i>diag</i> | selects if the triangular matrix unit-diagonal (FflasUnit/NoUnit)                     |
|         | <i>M</i>    | row dimension of A                                                                    |
|         | <i>N</i>    | column dimension of A                                                                 |
|         | <i>R</i>    | rank of the triangular matrix                                                         |
| in, out | <i>A</i>    | input/output matrix                                                                   |
|         | <i>lda</i>  | leading dimension of A                                                                |

**Todo** just one triangular fzero+fassign ?

**Todo** just one triangular fzero+fassign ?

**11.21.3.72 getEchelonForm() [1/2]**

```
template<class Field >
void getEchelonForm (
    const Field & F,
    const FFLAS::FFLAS_UPLO Uplo,
    const FFLAS::FFLAS_DIAG diag,
    const size_t M,
    const size_t N,
    const size_t R,
    const size_t * P,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::Element_ptr T,
    const size_t ldt,
    const bool OnlyNonZeroVectors = false,
    const FFPACK_LU_TAG LuTag = FfpackSlabRecursive) [inline]
```

Extracts a matrix in echelon form from a compact storage  $A=LU$  of rank  $R$  obtained by RowEchelonForm or ColumnEchelonForm.

Either L or U is in Echelon form (depending on Uplo) The echelon structure is defined by the first  $R$  values of the array  $P$ . row and column dimension of  $T$  are greater or equal to that of  $A$

## Parameters

|     |                           |                                                                                       |
|-----|---------------------------|---------------------------------------------------------------------------------------|
|     | $F$                       | base field                                                                            |
|     | $UpLo$                    | selects if the upper (FflasUpper) or lower (FflasLower) triangular matrix is returned |
|     | $diag$                    | selects if the echelon matrix has unit pivots (FflasUnit/NoUnit)                      |
|     | $M$                       | row dimension of T                                                                    |
|     | $N$                       | column dimension of T                                                                 |
|     | $R$                       | rank of the triangular matrix (how many rows/columns need to be copied)               |
|     | $P$                       | positions of the R pivots                                                             |
| in  | $A$                       | input matrix                                                                          |
|     | $lda$                     | leading dimension of A                                                                |
| out | $T$                       | output matrix                                                                         |
|     | $ldt$                     | leading dimension of T                                                                |
|     | <i>OnlyNonZeroVectors</i> | decides whether the last zero rows/columns should be ignored                          |
|     | $LuTag$                   | which factorized form (CUP/PLE if FfpackSlabRecursive, PLUQ if FfpackTileRecursive)   |

11.21.3.73 `getEchelonForm()` [2/2]

```
template<class Field >
void getEchelonForm (
    const Field & F,
    const FFLAS::FFLAS_UPLO Uplo,
    const FFLAS::FFLAS_DIAG diag,
    const size_t M,
    const size_t N,
    const size_t R,
    const size_t * P,
    typename Field::Element_ptr A,
    const size_t lda,
    const FFPACK_LU_TAG LuTag = FfpackSlabRecursive) [inline]
```

Cleans up a compact storage  $A=L\backslash U$  obtained by RowEchelonForm or ColumnEchelonForm to reveal an echelon form of rank R.

Either L or U is in Echelon form (depending on Uplo) The echelon structure is defined by the first R values of the array P.

## Parameters

|         |         |                                                                                       |
|---------|---------|---------------------------------------------------------------------------------------|
|         | $F$     | base field                                                                            |
|         | $UpLo$  | selects if the upper (FflasUpper) or lower (FflasLower) triangular matrix is returned |
|         | $diag$  | selects if the echelon matrix has unit pivots (FflasUnit/NoUnit)                      |
|         | $M$     | row dimension of A                                                                    |
|         | $N$     | column dimension of A                                                                 |
|         | $R$     | rank of the triangular matrix (how many rows/columns need to be copied)               |
|         | $P$     | positions of the R pivots                                                             |
| in, out | $A$     | input/output matrix                                                                   |
|         | $lda$   | leading dimension of A                                                                |
|         | $LuTag$ | which factorized form (CUP/PLE if FfpackSlabRecursive, PLUQ if FfpackTileRecursive)   |



**11.21.3.74 getEchelonTransform()**

```

template<class Field >
void getEchelonTransform (
    const Field & F,
    const FFLAS::FFLAS_UPLO Uplo,
    const FFLAS::FFLAS_DIAG diag,
    const size_t M,
    const size_t N,
    const size_t R,
    const size_t * P,
    const size_t * Q,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::Element_ptr T,
    const size_t ldt,
    const FFPACK_LU_TAG LuTag = FfpackSlabRecursive) [inline]

```

Extracts a transformation matrix to echelon form from a compact storage  $A=LU$  of rank  $R$  obtained by Row↔EchelonForm or ColumnEchelonForm.

If Uplo == FflasLower:  $T$  is  $N \times N$  (already allocated) such that  $A T = C$  is a transformation of  $A$  in Column echelon form Else  $T$  is  $M \times M$  (already allocated) such that  $T A = E$  is a transformation of  $A$  in Row Echelon form

**Parameters**

|     |              |                                                                                                         |
|-----|--------------|---------------------------------------------------------------------------------------------------------|
|     | <i>F</i>     | base field                                                                                              |
|     | <i>UpLo</i>  | Lower (FflasLower) means Transformation to Column Echelon Form, Upper (FflasUpper), to Row Echelon Form |
|     | <i>diag</i>  | selects if the echelon matrix has unit pivots (FflasUnit/NoUnit)                                        |
|     | <i>M</i>     | row dimension of A                                                                                      |
|     | <i>N</i>     | column dimension of A                                                                                   |
|     | <i>R</i>     | rank of the triangular matrix                                                                           |
|     | <i>P</i>     | permutation matrix                                                                                      |
| in  | <i>A</i>     | input matrix                                                                                            |
|     | <i>lda</i>   | leading dimension of A                                                                                  |
| out | <i>T</i>     | output matrix                                                                                           |
|     | <i>ldt</i>   | leading dimension of T                                                                                  |
|     | <i>LuTag</i> | which factorized form (CUP/PLE if FfpackSlabRecursive, PLUQ if FfpackTileRecursive)                     |

**11.21.3.75 getReducedEchelonForm() [1/2]**

```

template<class Field >
void getReducedEchelonForm (
    const Field & F,
    const FFLAS::FFLAS_UPLO Uplo,
    const size_t M,
    const size_t N,
    const size_t R,
    const size_t * P,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::Element_ptr T,

```

```
const size_t ldt,
const bool OnlyNonZeroVectors = false,
const FFPACK_LU_TAG LuTag = FfpackSlabRecursive) [inline]
```

Extracts a matrix in echelon form from a compact storage  $A=L\backslash U$  of rank  $R$  obtained by ReducedRowEchelonForm or ReducedColumnEchelonForm with transform = true.

Either  $L$  or  $U$  is in Echelon form (depending on Uplo) The echelon structure is defined by the first  $R$  values of the array  $P$ . row and column dimension of  $T$  are greater or equal to that of  $A$

#### Parameters

|    |                      |                                                                                       |
|----|----------------------|---------------------------------------------------------------------------------------|
|    | $F$                  | base field                                                                            |
|    | $UpLo$               | selects if the upper (FflasUpper) or lower (FflasLower) triangular matrix is returned |
|    | $diag$               | selects if the echelon matrix has unit pivots (FflasUnit/NoUnit)                      |
|    | $M$                  | row dimension of $T$                                                                  |
|    | $N$                  | column dimension of $T$                                                               |
|    | $R$                  | rank of the triangular matrix (how many rows/columns need to be copied)               |
|    | $P$                  | positions of the $R$ pivots                                                           |
| in | $A$                  | input matrix                                                                          |
|    | $lda$                | leading dimension of $A$                                                              |
|    | $ldt$                | leading dimension of $T$                                                              |
|    | $LuTag$              | which factorized form (CUP/PLE if FfpackSlabRecursive, PLUQ if FfpackTileRecursive)   |
|    | $OnlyNonZeroVectors$ | decides whether the last zero rows/columns should be ignored                          |

#### 11.21.3.76 getReducedEchelonForm() [2/2]

```
template<class Field >
void getReducedEchelonForm (
    const Field & F,
    const FFLAS::FFLAS_UPLO Uplo,
    const size_t M,
    const size_t N,
    const size_t R,
    const size_t * P,
    typename Field::Element_ptr A,
    const size_t lda,
    const FFPACK_LU_TAG LuTag = FfpackSlabRecursive) [inline]
```

Cleans up a compact storage  $A=L\backslash U$  of rank  $R$  obtained by ReducedRowEchelonForm or ReducedColumnEchelonForm with transform = true.

Either  $L$  or  $U$  is in Echelon form (depending on Uplo) The echelon structure is defined by the first  $R$  values of the array  $P$ .

#### Parameters

|  |        |                                                                                       |
|--|--------|---------------------------------------------------------------------------------------|
|  | $F$    | base field                                                                            |
|  | $UpLo$ | selects if the upper (FflasUpper) or lower (FflasLower) triangular matrix is returned |
|  | $diag$ | selects if the echelon matrix has unit pivots (FflasUnit/NoUnit)                      |
|  | $M$    | row dimension of $A$                                                                  |

## Parameters

|         |         |                                                                                     |
|---------|---------|-------------------------------------------------------------------------------------|
|         | $N$     | column dimension of A                                                               |
|         | $R$     | rank of the triangular matrix (how many rows/columns need to be copied)             |
|         | $P$     | positions of the R pivots                                                           |
| in, out | $A$     | input/output matrix                                                                 |
|         | $lda$   | leading dimension of A                                                              |
|         | $LuTag$ | which factorized form (CUP/PLE if FfpackSlabRecursive, PLUQ if FfpackTileRecursive) |

## 11.21.3.77 getReducedEchelonTransform()

```
template<class Field >
void getReducedEchelonTransform (
    const Field & F,
    const FFLAS::FFLAS_UPLO Uplo,
    const size_t M,
    const size_t N,
    const size_t R,
    const size_t * P,
    const size_t * Q,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::Element_ptr T,
    const size_t ldt,
    const FFPACK_LU_TAG LuTag = FfpackSlabRecursive) [inline]
```

Extracts a transformation matrix to echelon form from a compact storage  $A=LU$  of rank  $R$  obtained by Row↔EchelonForm or ColumnEchelonForm.

If Uplo == FflasLower:  $T$  is  $N \times N$  (already allocated) such that  $A T = C$  is a transformation of  $A$  in Column echelon form Else  $T$  is  $M \times M$  (already allocated) such that  $T A = E$  is a transformation of  $A$  in Row Echelon form

## Parameters

|     |         |                                                                                     |
|-----|---------|-------------------------------------------------------------------------------------|
|     | $F$     | base field                                                                          |
|     | $UpLo$  | selects Col (FflasLower) or Row (FflasUpper) Echelon Form                           |
|     | $diag$  | selects if the echelon matrix has unit pivots (FflasUnit/NoUnit)                    |
|     | $M$     | row dimension of A                                                                  |
|     | $N$     | column dimension of A                                                               |
|     | $R$     | rank of the triangular matrix                                                       |
|     | $P$     | permutation matrix                                                                  |
| in  | $A$     | input matrix                                                                        |
|     | $lda$   | leading dimension of A                                                              |
| out | $T$     | output matrix                                                                       |
|     | $ldt$   | leading dimension of T                                                              |
|     | $LuTag$ | which factorized form (CUP/PLE if FfpackSlabRecursive, PLUQ if FfpackTileRecursive) |

### 11.21.3.78 PLUQtoEchelonPermutation()

```
void PLUQtoEchelonPermutation (
    const size_t N,
    const size_t R,
    const size_t * P,
    size_t * outPerm) [inline]
```

Auxiliary routine: determines the permutation that changes a PLUQ decomposition into a echelon form revealing PLUQ decomposition.

### 11.21.3.79 LTBruhatGen()

```
template<class Field >
size_t LTBruhatGen (
    const Field & Fi,
    const FFLAS::FFLAS_DIAG diag,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t * Q) [inline]
```

LTBruhatGen Suppose A is Left Triangular Matrix This procedure computes the Bruhat Representation of A and return the rank of A.

#### Parameters

|             |                                                |
|-------------|------------------------------------------------|
| <i>Fi</i>   | base Field                                     |
| <i>diag</i> |                                                |
| <i>N</i>    | size of A                                      |
| <i>A</i>    | the matrix we search the Bruhat representation |
| <i>lda</i>  | the leading dimension of A                     |
| <i>P</i>    | a permutation matrix                           |
| <i>Q</i>    | a permutation matrix                           |

### 11.21.3.80 getLTBruhatGen() [1/2]

```
template<class Field >
void getLTBruhatGen (
    const Field & Fi,
    const size_t N,
    const size_t r,
    const size_t * P,
    const size_t * Q,
    typename Field::Element_ptr R,
    const size_t ldr) [inline]
```

GetLTBruhatGen This procedure Computes the Rank Revealing Matrix based on the Bruhta representation of a Matrix.

## Parameters

|            |                                                        |
|------------|--------------------------------------------------------|
| <i>Fi</i>  | base Field                                             |
| <i>N</i>   | size of the matrix                                     |
| <i>r</i>   | the rank of the matrix                                 |
| <i>P</i>   | a permutation matrix                                   |
| <i>Q</i>   | a permutation matrix                                   |
| <i>R</i>   | the matrix that will contain the rank revealing matrix |
| <i>ldr</i> | the leading fimension of R                             |

## 11.21.3.81 getLTBruhatGen() [2/2]

```
template<class Field >
void getLTBruhatGen (
    const Field & Fi,
    const FFLAS::FFLAS_UPLO Uplo,
    const FFLAS::FFLAS_DIAG diag,
    const size_t N,
    const size_t r,
    const size_t * P,
    const size_t * Q,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::Element_ptr T,
    const size_t ldt) [inline]
```

GetLTBruhatGen This procedure computes the matrix L or U f the Bruhat Representation Suppose that A is the bruhat representation of a matrix.

## Parameters

|             |                                       |
|-------------|---------------------------------------|
| <i>Fi</i>   | base Field                            |
| <i>Uplo</i> | choose if the procedure return L or U |
| <i>diag</i> |                                       |
| <i>N</i>    | size of A                             |
| <i>r</i>    | rank of A                             |
| <i>P</i>    | permutaion matrix                     |
| <i>Q</i>    | permutation matrix                    |
| <i>A</i>    | a bruhat representation               |
| <i>lda</i>  | leading dimension of A                |
| <i>T</i>    | matrix that will contains L or U      |
| <i>ldt</i>  | leading dimension of T                |

## 11.21.3.82 LTQSorter()

```
size_t LTQSorter (
    const size_t N,
    const size_t r,
    const size_t * P,
    const size_t * Q) [inline]
```

LTQSorter This procedure computes the order of quasiseparability of a matrix.

## Parameters

|     |                    |
|-----|--------------------|
| $N$ | size of the matrix |
| $r$ | rank of the matrix |
| $P$ | permutation matrix |
| $Q$ | permutation matrix |

**11.21.3.83 CompressToBlockBiDiagonal()**

```
template<class Field >
size_t CompressToBlockBiDiagonal (
    const Field & Fi,
    const FFLAS::FFLAS_UPLO Uplo,
    size_t N,
    size_t s,
    size_t r,
    const size_t * P,
    const size_t * Q,
    typename Field::Element_ptr A,
    size_t lda,
    typename Field::Element_ptr X,
    size_t ldx,
    size_t * K,
    size_t * M,
    size_t * T) [inline]
```

**CompressToBlockBiDiagonal** This procedure compress a compact representation of a row echelon form or column echelon form.

## Parameters

|        |                                                   |
|--------|---------------------------------------------------|
| $Fi$   | base Field                                        |
| $Uplo$ | chosse if the procedure is based on row or column |
| $N$    | size of the matrix                                |
| $s$    | order of qausiseparability                        |
| $r$    | rank                                              |
| $P$    | permutation matrix                                |
| $Q$    | permutation matrix                                |
| $A$    | the matrix to compact                             |
| $lda$  | leading dimension of A                            |
| $X$    | matrix that will stock the representation         |
| $ldx$  | leading dimension of X                            |
| $K$    | stock the position of the blocks in A             |
| $M$    | permutation matrix                                |
| $T$    | stock the operation done in the procedure         |

**11.21.3.84 ExpandBlockBiDiagonalToBruhat()**

```
template<class Field >
void ExpandBlockBiDiagonalToBruhat (
```

```

const Field & Fi,
const FFLAS::FFLAS_UPLO Uplo,
size_t N,
size_t s,
size_t r,
typename Field::Element_ptr A,
size_t lda,
typename Field::Element_ptr X,
size_t ldx,
size_t NbBlocks,
size_t * K,
size_t * M,
size_t * T) [inline]

```

**ExpandBlockBiDiagonal** This procedure expand a compact representation of a row echelon form or column echelon form.

#### Parameters

|             |                                                        |
|-------------|--------------------------------------------------------|
| <i>Fi</i>   | base Field                                             |
| <i>Uplo</i> | chosse if the procedure is based on row or column      |
| <i>N</i>    | size of the matrix                                     |
| <i>s</i>    | order of qausiseparability                             |
| <i>r</i>    | rank                                                   |
| <i>A</i>    | the matrix that will sotck the expanded representation |
| <i>lda</i>  | leading dimension of A                                 |
| <i>X</i>    | matrix to expand                                       |
| <i>ldx</i>  | leading dimension of X                                 |
| <i>K</i>    | stock the position of the blocks in A                  |
| <i>M</i>    | permutation matrix                                     |
| <i>T</i>    | stock the operation done in the procedure              |

#### 11.21.3.85 Bruhat2EchelonPermutation()

```

void Bruhat2EchelonPermutation (
    size_t N,
    size_t R,
    const size_t * P,
    const size_t * Q,
    size_t * M) [inline]

```

**Bruhat2EchelonPermutation (N,R,P,Q)** Compute M such that LM or MU is in echelon form where L or U are factors of the Bruhat Rpresentation.

#### Parameters

|     |          |                           |
|-----|----------|---------------------------|
| in  | <i>N</i> | size of the matrix        |
| in  | <i>R</i> | rank                      |
| in  | <i>P</i> | permutation Matrix        |
| in  | <i>Q</i> | permutation Matrix        |
| out | <i>M</i> | output permutation matrix |

**11.21.3.86 TInverter()** [1/2]

```
size_t * TInverter (
    size_t * T,
    size_t r)
```

**11.21.3.87 ComputeRPermutation()** [1/2]

```
template<class Field >
void ComputeRPermutation (
    const Field & Fi,
    size_t N,
    size_t r,
    const size_t * P,
    const size_t * Q,
    size_t * R,
    size_t * MU,
    size_t * ML)
```

**11.21.3.88 productBruhatxTS()** [1/2]

```
template<class Field >
void productBruhatxTS (
    const Field & Fi,
    size_t N,
    size_t s,
    size_t r,
    const size_t * P,
    const size_t * Q,
    const typename Field::Element_ptr Xu,
    size_t ldu,
    size_t NbBlocksU,
    size_t * Ku,
    size_t * Tu,
    size_t * MU,
    const typename Field::Element_ptr Xl,
    size_t ldl,
    size_t NbBlocksL,
    size_t * Kl,
    size_t * Tl,
    size_t * ML,
    typename Field::Element_ptr B,
    size_t t,
    size_t ldb,
    typename Field::Element_ptr C,
    size_t ldc)
```

productBruhatxTS Comput the product between the CRE compact representation of a matrix A and B a tall matrix



**11.21.3.89 LQUPtoInverseOfFullRankMinor()** [1/2]

```
template<class Field >
Field::Element_ptr LQUPtoInverseOfFullRankMinor (
    const Field & F,
    const size_t rank,
    typename Field::Element_ptr A_factors,
    const size_t lda,
    const size_t * QtPointer,
    typename Field::Element_ptr X,
    const size_t ldx)
```

LQUPtoInverseOfFullRankMinor.

Suppose A has been factorized as L.Q.U.P, with rank r. Then Qt.A.Pt has an invertible leading principal  $r \times r$  submatrix This procedure efficiently computes the inverse of this minor and puts it into X.

**Note**

It changes the lower entries of A\_factors in the process (NB: unless A was nonsingular and square)

**Parameters**

|                  |                                                            |
|------------------|------------------------------------------------------------|
| <i>F</i>         | base field                                                 |
| <i>rank</i>      | rank of the matrix.                                        |
| <i>A_factors</i> | matrix containing the L and U entries of the factorization |
| <i>lda</i>       | leading dimension of A                                     |
| <i>QtPointer</i> | theLQUP->getQ()->getPointer() (note: getQ returns Qt!)     |
| <i>X</i>         | desired location for output                                |
| <i>ldx</i>       | leading dimension of X                                     |

**11.21.3.90 RandomNullSpaceVector()** [2/3]

```
template<class Field >
void RandomNullSpaceVector (
    const Field & F,
    const FFLAS::FFLAS_SIDE Side,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr X,
    const size_t incX)
```

Solve  $LX = B$  or  $XL = B$  in place.

L is  $M \times M$  if Side == FFLAS::FflasLeft and  $N \times N$  if Side == FFLAS::FflasRight, B is  $M \times N$ . Only the R non trivial column of L are stored in the  $M \times R$  matrix L Requirement : so that L could be expanded in-place Computes a vector of the Left/Right nullspace of the matrix A.

**Parameters**

|  |          |                        |
|--|----------|------------------------|
|  | <i>F</i> | The computation domain |
|--|----------|------------------------|

## Parameters

|                |             |                                                                                  |
|----------------|-------------|----------------------------------------------------------------------------------|
|                | <i>Side</i> | decides whether it computes the left (FflasLeft) or right (FflasRight) nullspace |
|                | <i>M</i>    | number of rows                                                                   |
|                | <i>N</i>    | number of columns                                                                |
| <i>in, out</i> | <i>A</i>    | input matrix of dimension M x N, A is modified to its LU version                 |
|                | <i>lda</i>  | leading dimension of A                                                           |
| <i>out</i>     | <i>X</i>    | output vector                                                                    |
|                | <i>incX</i> | increment of X                                                                   |

**11.21.3.91 solveLB()** [1/2]

```
template<class Field >
void solveLB (
    const Field & F,
    const FFLAS::FFLAS_SIDE Side,
    const size_t M,
    const size_t N,
    const size_t R,
    typename Field::Element_ptr L,
    const size_t ldl,
    const size_t * Q,
    typename Field::Element_ptr B,
    const size_t ldb)
```

**11.21.3.92 solveLB2()** [1/2]

```
template<class Field >
void solveLB2 (
    const Field & F,
    const FFLAS::FFLAS_SIDE Side,
    const size_t M,
    const size_t N,
    const size_t R,
    typename Field::Element_ptr L,
    const size_t ldl,
    const size_t * Q,
    typename Field::Element_ptr B,
    const size_t ldb)
```

**11.21.3.93 TInverter()** [2/2]

```
size_t * TInverter (
    const size_t * T,
    size_t r) [inline]
```

**11.21.3.94 ComputeRPermutation()** [2/2]

```
template<class Field >
void ComputeRPermutation (
    const Field & Fi,
    size_t N,
    size_t r,
    const size_t * P,
    const size_t * Q,
    size_t * R,
    const size_t * MU,
    const size_t * ML) [inline]
```

**11.21.3.95 expandLCRE()**

```
template<class Field >
Field::Element_ptr expandLCRE (
    const Field & Fi,
    size_t N,
    size_t s,
    size_t r,
    size_t * R,
    size_t i,
    typename Field::ConstElement_ptr Xu,
    size_t ldu,
    size_t NbBlocksU,
    const size_t * Ku,
    const size_t * Tuinv,
    typename Field::ConstElement_ptr Xl,
    size_t ldl,
    size_t NbBlocksL,
    const size_t * Kl,
    const size_t * Tlinv,
    typename Field::Element_ptr CRE,
    size_t ldcre) [inline]
```

Expands an anti-diagonal block of a left triangular matrix from its compact Bruhat representation.

**11.21.3.96 productBruhatxTS()** [2/2]

```
template<class Field >
void productBruhatxTS (
    const Field & Fi,
    size_t N,
    size_t s,
    size_t r,
    size_t t,
    const size_t * P,
    const size_t * Q,
    typename Field::ConstElement_ptr Xu,
    size_t ldu,
    size_t NbBlocksU,
    const size_t * Ku,
    const size_t * Tu,
```

```

const size_t * MU,
typename Field::ConstElement_ptr Xl,
size_t ldl,
size_t NbBlocksL,
const size_t * Kl,
const size_t * Tl,
const size_t * ML,
typename Field::Element_ptr B,
size_t ldb,
const typename Field::Element beta,
typename Field::Element_ptr D,
size_t ldd) [inline]

```

Compute the product of a left-triangular quasi-separable matrix A, represented by a compact Bruhat generator, with a dense rectangular matrix B:  $C \leftarrow A \times B + \text{beta}C$ .

#### Parameters

|         |               |                                                                                                     |
|---------|---------------|-----------------------------------------------------------------------------------------------------|
|         | $F$           | the base field                                                                                      |
|         | $N$           | the order of A                                                                                      |
|         | $s$           | the order of quasiseparability of A                                                                 |
|         | $r$           | the number of pivots in the left-triangular par of the rank profile matrix of A                     |
|         | $t$           | the number of columns of B                                                                          |
|         | $P$           | the row indices of the pivots of A                                                                  |
|         | $Q$           | the column indices of the pivots of A                                                               |
|         | $Xu$          | the compact storage of U: $Du$ blocks in the first $s$ rows, $Su$ blocks in the last $s$ rows       |
|         | $ldxu$        | the leading dimension of $Xu$                                                                       |
|         | $NbBlocksU$   | the number of diagonal blocks in the compact storage of U                                           |
|         | $Ku$          | the list of starting column positions for each block of the storage of U                            |
|         | $Tu$          | the folding matrix for the compact storage of U: $Du + TuSu$ is in row echelon form                 |
|         | $Mu$          | a permutation matrix such that $Mu(Du + TuSu)$ is the U factor of the Bruhat generator              |
|         | $Xl$          | the compact storage of L: $Dl$ blocks in the first $s$ columns, $Sl$ blocks in the last $s$ columns |
|         | $ldxl$        | the leading dimension of $Xl$                                                                       |
|         | $NbBlocksL$   | the number of diagonal blocks in the compact storage of L                                           |
|         | $Kl$          | the list of starting row positions for each block of the storage of L                               |
|         | $Tl$          | the folding matrix for the compact storage of L: $Dl + SlTl$ is in column echelon form              |
|         | $Ml$          | a permutation matrix such that $(Dl + SlTl)Ml$ is the L factor of the Bruhat generator              |
|         | $B$           | an $N \times t$ dense matrix                                                                        |
|         | $ldb$         | leading dimension of B                                                                              |
|         | $\text{beta}$ | scaling constant                                                                                    |
| in, out | $C$           | output matrix                                                                                       |
|         | $ldc$         | leading dimension of C                                                                              |

**Bibliography** Pernet C. and Storjohann A. *Time and space efficient generators for quasiseparable matrices*, JSC (85), 2018, doi:10.1016/j.jsc.2017.07.010

**11.21.3.97 Danilevski()**

```
template<class Field , class Polynomial >
std::list< Polynomial > & Danilevski (
    const Field & F,
    std::list< Polynomial > & charp,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda)
```

**11.21.3.98 buildMatrix()**

```
template<class Field >
Field::Element_ptr buildMatrix (
    const Field & F,
    typename Field::ConstElement_ptr E,
    typename Field::ConstElement_ptr C,
    const size_t lda,
    const size_t * B,
    const size_t * T,
    const size_t me,
    const size_t mc,
    const size_t lambda,
    const size_t mu)
```

**Bug** is this :

**11.21.3.99 CharPoly() [4/8]**

```
FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr CharPoly (
    const FFPACK::RNSInteger< FFPACK::rns_double > & F,
    typename FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr charp,
    const size_t N,
    typename FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr A,
    const size_t lda,
    Givaro::ZRing< Givaro::Integer >::RandIter & G,
    const FFPACK_CHARPOLY_TAG CharpTag,
    size_t degree) [inline]
```

**11.21.3.100 CharPoly() [5/8]**

```
template<>
Givaro::Poly1Dom< Givaro::ZRing< Givaro::Integer > >::Element & CharPoly (
    const Givaro::Poly1Dom< Givaro::ZRing< Givaro::Integer > > & R,
    Givaro::Poly1Dom< Givaro::ZRing< Givaro::Integer > >::Element & charp,
    const size_t N,
    Givaro::Integer * A,
    const size_t lda,
    Givaro::ZRing< Givaro::Integer >::RandIter & G,
    const FFPACK_CHARPOLY_TAG CharpTag,
    size_t degree) [inline]
```

**11.21.3.101 Det()** [3/6]

```
template<class PSHelper >
FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr & Det (
    const FFPACK::RNSInteger< FFPACK::rns_double > & F,
    typename FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr & det,
    const size_t N,
    typename FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr A,
    const size_t lda,
    const PSHelper & psH) [inline]
```

**11.21.3.102 Det()** [4/6]

```
template<class PSHelper >
Givaro::Integer & Det (
    const Givaro::ZRing< Givaro::Integer > & F,
    Givaro::Integer & det,
    const size_t N,
    Givaro::Integer * A,
    const size_t lda,
    const PSHelper & psH,
    size_t * P,
    size_t * Q) [inline]
```

**11.21.3.103 fsytrf\_BC\_Crout()**

```
template<class Field >
bool fsytrf_BC_Crout (
    const Field & F,
    const FFLAS::FFLAS_UPLO UpLo,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr Dinv,
    const size_t incDinv) [inline]
```

**11.21.3.104 fsytrf\_BC\_RL()**

```
template<class Field >
size_t fsytrf_BC_RL (
    const Field & F,
    const FFLAS::FFLAS_UPLO UpLo,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr Dinv,
    const size_t incDinv) [inline]
```

**11.21.3.105 fsytrf\_UP\_RPM\_BC\_RL()**

```
template<class Field >
size_t fsytrf_UP_RPM_BC_RL (
    const Field & F,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr Dinv,
    const size_t incDinv,
    size_t * P) [inline]
```

**11.21.3.106 fsytrf\_LOW\_RPM\_BC\_Crout()**

```
template<class Field >
size_t fsytrf_LOW_RPM_BC_Crout (
    const Field & F,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr Dinv,
    const size_t incDinv,
    size_t * P) [inline]
```

**11.21.3.107 fsytrf\_UP\_RPM\_BC\_Crout()**

```
template<class Field >
size_t fsytrf_UP_RPM_BC_Crout (
    const Field & F,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr Dinv,
    const size_t incDinv,
    size_t * P) [inline]
```

**11.21.3.108 fsytrf\_UP\_RPM()**

```
template<class Field >
size_t fsytrf_UP_RPM (
    const Field & Fi,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr Dinv,
    const size_t incDinv,
    size_t * P,
    size_t BCThreshold) [inline]
```

MathP <- [ [ I ] x P1 | [ [ L (N1+R2) ] [ P2^T ] ] ] x [ P3^T ] [ ----- | ---- ] [ [ Q2^T ]

Changing [ U1 V1 | E1 E21 E22 ] into [ U1 E11 E12 V1 E\* E\* ] [ 0 | L2 \ U2 V21 V22 ] [ U4 V41 0 V42 V43 ] [ 0 | M2 0 0 ] [ U3 0 0 V3 ] [ ----- ] [ 0 0 0 ] [ 0 | H1 H21 H22 ] [ 0 | U3 V3 ] [ 0 | 0 ] where U4 is the 2R2 x 2R2 matrix formed by interleaving U2, L2^T and H1

**11.21.3.109 fsytrf\_nonunit() [2/3]**

```

template<class Field >
bool fsytrf_nonunit (
    const Field & F,
    const FFLAS::FFLAS_UPLO UpLo,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr Dinv,
    const size_t incDinv,
    FFLAS::ParSeqHelper::Sequential seq,
    size_t threshold) [inline]

```

**11.21.3.110 fsytrf\_nonunit() [3/3]**

```

template<class Field , class Cut , class Param >
bool fsytrf_nonunit (
    const Field & F,
    const FFLAS::FFLAS_UPLO UpLo,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr Dinv,
    const size_t incDinv,
    FFLAS::ParSeqHelper::Parallel< Cut, Param > par,
    size_t threshold) [inline]

```

**11.21.3.111 fsytrf\_RPM()**

```

template<class Field >
size_t fsytrf_RPM (
    const Field & F,
    const FFLAS::FFLAS_UPLO UpLo,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t threshold) [inline]

```

**11.21.3.112 getTridiagonal()**

```

template<class Field >
void getTridiagonal (
    const Field & F,
    const size_t N,
    const size_t R,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    size_t * P,
    typename Field::Element_ptr T,
    const size_t ldt) [inline]

```



**11.21.3.113 LUdivine\_gauss()** [1/2]

```
template<class Field >
size_t LUdivine_gauss (
    const Field & F,
    const FFLAS::FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t * Q,
    const FFPACK::FFPACK_LU_TAG LuTag) [inline]
```

**11.21.3.114 LUdivine\_small()** [1/2]

```
template<class Field >
size_t LUdivine_small (
    const Field & F,
    const FFLAS::FFLAS_DIAG Diag,
    const FFLAS::FFLAS_TRANSPOSE trans,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t * Q,
    const FFPACK::FFPACK_LU_TAG LuTag) [inline]
```

**11.21.3.115 LUdivine()** [2/4]

```
template<class Field >
size_t LUdivine (
    const Field & F,
    const FFLAS::FFLAS_DIAG Diag,
    const FFLAS::FFLAS_TRANSPOSE trans,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t * Q,
    const FFPACK::FFPACK_LU_TAG LuTag,
    const size_t cutoff) [inline]
```

**Todo** std::swap ?

**11.21.3.116 LUdivine() [3/4]**

```

template<>
size_t LUdivine (
    const Givaro::Modular< Givaro::Integer > & F,
    const FFLAS::FFLAS_DIAG Diag,
    const FFLAS::FFLAS_TRANSPOSE trans,
    const size_t M,
    const size_t N,
    typename Givaro::Integer * A,
    const size_t lda,
    size_t * P,
    size_t * Q,
    const FFPACK::FFPACK_LU_TAG LuTag,
    const size_t cutoff) [inline]

```

**11.21.3.117 MonotonicCompress()**

```

template<class Field >
void MonotonicCompress (
    const Field & F,
    const FFLAS::FFLAS_SIDE Side,
    const size_t M,
    typename Field::Element_ptr A,
    const size_t lda,
    const size_t incA,
    const size_t * MathP,
    const size_t R,
    const size_t maxpiv,
    const size_t rowstomove,
    const std::vector< bool > & ispiv) [inline]

```

**11.21.3.118 MonotonicCompressMorePivots()**

```

template<class Field >
void MonotonicCompressMorePivots (
    const Field & F,
    const FFLAS::FFLAS_SIDE Side,
    const size_t M,
    typename Field::Element_ptr A,
    const size_t lda,
    const size_t incA,
    const size_t * MathP,
    const size_t R,
    const size_t rowstomove,
    const size_t lenP) [inline]

```

**11.21.3.119 MonotonicCompressCycles()**

```

template<class Field >
void MonotonicCompressCycles (
    const Field & F,

```

```

    const FFLAS::FFLAS_SIDE Side,
    const size_t M,
    typename Field::Element_ptr A,
    const size_t lda,
    const size_t incA,
    const size_t * MathP,
    const size_t lenP) [inline]

```

### 11.21.3.120 MonotonicExpand()

```

template<class Field >
void MonotonicExpand (
    const Field & F,
    const FFLAS::FFLAS_SIDE Side,
    const size_t M,
    typename Field::Element_ptr A,
    const size_t lda,
    const size_t incA,
    const size_t * MathP,
    const size_t R,
    const size_t maxpiv,
    const size_t rowstomove,
    const std::vector< bool > & ispiv)

```

### 11.21.3.121 applyP\_block()

```

template<class Field >
void applyP_block (
    const Field & F,
    const FFLAS::FFLAS_SIDE Side,
    const FFLAS::FFLAS_TRANSPOSE Trans,
    const size_t M,
    const size_t ibeg,
    const size_t iend,
    typename Field::Element_ptr A,
    const size_t lda,
    const size_t * P) [inline]

```

### 11.21.3.122 doApplyS()

```

template<class Field >
void doApplyS (
    const Field & F,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr tmp,
    const size_t width,
    const size_t M2,
    const size_t R1,
    const size_t R2,
    const size_t R3,
    const size_t R4) [inline]

```

**11.21.3.123 MatrixApplyS() [1/3]**

```
template<class Field >
void MatrixApplyS (
    const Field & F,
    typename Field::Element_ptr A,
    const size_t lda,
    const size_t width,
    const size_t M2,
    const size_t R1,
    const size_t R2,
    const size_t R3,
    const size_t R4) [inline]
```

**11.21.3.124 MatrixApplyS() [2/3]**

```
template<class Field >
void MatrixApplyS (
    const Field & F,
    typename Field::Element_ptr A,
    const size_t lda,
    const size_t width,
    const size_t M2,
    const size_t R1,
    const size_t R2,
    const size_t R3,
    const size_t R4,
    const FFLAS::ParSeqHelper::Sequential seq) [inline]
```

**11.21.3.125 MatrixApplyS() [3/3]**

```
template<class Field , class Cut , class Param >
void MatrixApplyS (
    const Field & F,
    typename Field::Element_ptr A,
    const size_t lda,
    const size_t width,
    const size_t M2,
    const size_t R1,
    const size_t R2,
    const size_t R3,
    const size_t R4,
    const FFLAS::ParSeqHelper::Parallel< Cut, Param > par) [inline]
```

**11.21.3.126 PermApplyS()**

```
template<class T >
void PermApplyS (
    T * A,
    const size_t lda,
    const size_t width,
    const size_t M2,
    const size_t R1,
    const size_t R2,
    const size_t R3,
    const size_t R4) [inline]
```

**11.21.3.127 doApplyT()**

```

template<class Field >
void doApplyT (
    const Field & F,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr tmp,
    const size_t width,
    const size_t N2,
    const size_t R1,
    const size_t R2,
    const size_t R3,
    const size_t R4) [inline]

```

**11.21.3.128 MatrixApplyT() [1/3]**

```

template<class Field >
void MatrixApplyT (
    const Field & F,
    typename Field::Element_ptr A,
    const size_t lda,
    const size_t width,
    const size_t N2,
    const size_t R1,
    const size_t R2,
    const size_t R3,
    const size_t R4) [inline]

```

**11.21.3.129 MatrixApplyT() [2/3]**

```

template<class Field >
void MatrixApplyT (
    const Field & F,
    typename Field::Element_ptr A,
    const size_t lda,
    const size_t width,
    const size_t N2,
    const size_t R1,
    const size_t R2,
    const size_t R3,
    const size_t R4,
    const FFLAS::ParSeqHelper::Sequential seq) [inline]

```

**11.21.3.130 MatrixApplyT() [3/3]**

```

template<class Field , class Cut , class Param >
void MatrixApplyT (
    const Field & F,
    typename Field::Element_ptr A,
    const size_t lda,
    const size_t width,

```

```

const size_t N2,
const size_t R1,
const size_t R2,
const size_t R3,
const size_t R4,
const FFLAS::ParSeqHelper::Parallel< Cut, Param > par) [inline]

```

### 11.21.3.131 PermApplyT()

```

template<class T >
void PermApplyT (
    T * A,
    const size_t lda,
    const size_t width,
    const size_t N2,
    const size_t R1,
    const size_t R2,
    const size_t R3,
    const size_t R4) [inline]

```

### 11.21.3.132 composePermutationsLLL()

```

void composePermutationsLLL (
    size_t * P1,
    const size_t * P2,
    const size_t R,
    const size_t N) [inline]

```

Computes  $P1 \times \text{Diag}(I_R, P2)$  where  $P1$  is a LAPACK and  $P2$  a LAPACK permutation and store the result in  $P1$  as a LAPACK permutation.

#### Parameters

|         |      |                                  |
|---------|------|----------------------------------|
| in, out | $P1$ | a LAPACK permutation of size N   |
|         | $P2$ | a LAPACK permutation of size N-R |

### 11.21.3.133 composePermutationsLLM()

```

void composePermutationsLLM (
    size_t * MathP,
    const size_t * P1,
    const size_t * P2,
    const size_t R,
    const size_t N) [inline]

```

Computes  $P1 \times \text{Diag}(I_R, P2)$  where  $P1$  is a LAPACK and  $P2$  a LAPACK permutation and store the result in  $\text{MathP}$  as a  $\text{MathPermutation}$  format.

#### Parameters

|     |  |  |
|-----|--|--|
| out |  |  |
|-----|--|--|

a  $\text{MathPermutation}$  of size N

## Parameters

|           |                                  |
|-----------|----------------------------------|
| <i>P1</i> | a LAPACK permutation of size N   |
| <i>P2</i> | a LAPACK permutation of size N-R |

**11.21.3.134 composePermutationsMLM()**

```
void composePermutationsMLM (
    size_t * MathP1,
    const size_t * P2,
    const size_t R,
    const size_t N) [inline]
```

Computes MathP1 x Diag (I\_R, P2) where MathP1 is a MathPermutation and P2 a LAPACK permutation and store the result in MathP1 as a MathPermutation format.

## Parameters

|         |               |                                  |
|---------|---------------|----------------------------------|
| in, out | <i>MathP1</i> | a MathPermutation of size N      |
|         | <i>P2</i>     | a LAPACK permutation of size N-R |

**11.21.3.135 cyclic\_shift\_mathPerm()**

```
void cyclic_shift_mathPerm (
    size_t * P,
    const size_t s) [inline]
```

**11.21.3.136 cyclic\_shift\_row\_col()** [1/2]

```
template<class Field >
void cyclic_shift_row_col (
    const Field & F,
    typename Field::Element_ptr A,
    size_t m,
    size_t n,
    size_t lda) [inline]
```

**11.21.3.137 cyclic\_shift\_row()** [1/3]

```
template<class Field >
void cyclic_shift_row (
    const Field & F,
    typename Field::Element_ptr A,
    size_t m,
    size_t n,
    size_t lda) [inline]
```

**11.21.3.138 cyclic\_shift\_row()** [2/3]

```
template<typename T >
void cyclic_shift_row (
    const RNSIntegerMod< T > & F,
    typename T::Element_ptr A,
    size_t m,
    size_t n,
    size_t lda) [inline]
```

**11.21.3.139 cyclic\_shift\_col()** [1/3]

```
template<class Field >
void cyclic_shift_col (
    const Field & F,
    typename Field::Element_ptr A,
    size_t m,
    size_t n,
    size_t lda) [inline]
```

**11.21.3.140 cyclic\_shift\_col()** [2/3]

```
template<typename T >
void cyclic_shift_col (
    const RNSIntegerMod< T > & F,
    typename T::Element_ptr A,
    size_t m,
    size_t n,
    size_t lda) [inline]
```

**11.21.3.141 PLUQ\_basecaseV3()**

```
template<class Field >
size_t PLUQ_basecaseV3 (
    const Field & Fi,
    const FFLAS::FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    typename Field::Element * A,
    const size_t lda,
    size_t * P,
    size_t * Q) [inline]
```

**11.21.3.142 PLUQ\_basecaseV2()**

```
template<class Field >
size_t PLUQ_basecaseV2 (
    const Field & Fi,
    const FFLAS::FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    typename Field::Element * A,
    const size_t lda,
    size_t * P,
    size_t * Q) [inline]
```



**11.21.3.143 PLUQ\_basecaseCrout()**

```
template<class Field >
size_t PLUQ_basecaseCrout (
    const Field & Fi,
    const FFLAS::FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t * Q) [inline]
```

**11.21.3.144 \_PLUQ()**

```
template<class Field >
size_t _PLUQ (
    const Field & Fi,
    const FFLAS::FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t * Q,
    size_t BCThreshold) [inline]
```

**11.21.3.145 PLUQ() [4/6]**

```
template<class Cut , class Param >
size_t PLUQ (
    const Givaro::Modular< Givaro::Integer > & F,
    const FFLAS::FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    typename Givaro::Integer * A,
    const size_t lda,
    size_t * P,
    size_t * Q,
    size_t BCThreshold,
    FFLAS::ParSeqHelper::Parallel< Cut, Param > & PSHelper) [inline]
```

**11.21.3.146 threads\_fgemm()**

```
template<class Field >
void threads_fgemm (
    const size_t m,
    const size_t n,
    const size_t r,
    int nbthreads,
    size_t * W1,
    size_t * W2,
    size_t * W3,
    size_t gamma)
```

**11.21.3.147 threads\_ftrsm()**

```
template<class Field >
void threads_ftrsm (
    const size_t m,
    const size_t n,
    int nbthreads,
    size_t * t1,
    size_t * t2)
```

**11.21.3.148 PLUQ() [5/6]**

```
template<class Field >
size_t PLUQ (
    const Field & Fi,
    const FFLAS::FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t * Q,
    const FFLAS::ParSeqHelper::Parallel< FFLAS::CuttingStrategy::Recursive, FFLAS::StrategyParameter
> & PSHelper) [inline]
```

**11.21.3.149 fflas\_const\_cast() [1/3]**

```
template<>
rns_double_elt_ptr fflas_const_cast (
    rns_double_elt_cstptr x) [inline]
```

**11.21.3.150 fflas\_const\_cast() [2/3]**

```
template<>
rns_double_elt_cstptr fflas_const_cast (
    rns_double_elt_ptr x) [inline]
```

**11.21.3.151 cyclic\_shift\_row\_col() [2/2]**

```
template<typename Base_t >
void cyclic_shift_row_col (
    Base_t * A,
    size_t m,
    size_t n,
    size_t lda)
```

**11.21.3.152 cyclic\_shift\_row()** [3/3]

```
template INST_OR_DECL void cyclic_shift_row (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    FFLAS_ELT * A,
    size_t m,
    size_t n,
    size_t lda)
```

**11.21.3.153 cyclic\_shift\_col()** [3/3]

```
template INST_OR_DECL void cyclic_shift_col (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    FFLAS_ELT * A,
    size_t m,
    size_t n,
    size_t lda)
```

**11.21.3.154 applyP()** [4/4]

```
template INST_OR_DECL void applyP (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS::FFLAS_SIDE Side,
    const FFLAS::FFLAS_TRANSPOSE Trans,
    const size_t M,
    const size_t ibeg,
    const size_t iend,
    FFLAS_ELT * A,
    const size_t lda,
    const size_t * P)
```

**11.21.3.155 fgetrs()** [3/4]

```
template INST_OR_DECL void fgetrs (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS::FFLAS_SIDE Side,
    const size_t M,
    const size_t N,
    const size_t R,
    FFLAS_ELT * A,
    const size_t lda,
    const size_t * P,
    const size_t * Q,
    FFLAS_ELT * B,
    const size_t ldb,
    int * info)
```

**11.21.3.156 fgetrs()** [4/4]

```
template INST_OR_DECL FFLAS_ELT * fgetrs (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS::FFLAS_SIDE Side,
    const size_t M,
    const size_t N,
    const size_t NRHS,
    const size_t R,
    FFLAS_ELT * A,
    const size_t lda,
    const size_t * P,
    const size_t * Q,
    FFLAS_ELT * X,
    const size_t ldx,
    const FFLAS_ELT * B,
    const size_t ldb,
    int * info)
```

**11.21.3.157 fgesv()** [3/4]

```
template INST_OR_DECL size_t fgesv (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS::FFLAS_SIDE Side,
    const size_t M,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda,
    FFLAS_ELT * B,
    const size_t ldb,
    int * info)
```

**11.21.3.158 fgesv()** [4/4]

```
template INST_OR_DECL size_t fgesv (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS::FFLAS_SIDE Side,
    const size_t M,
    const size_t N,
    const size_t NRHS,
    FFLAS_ELT * A,
    const size_t lda,
    FFLAS_ELT * X,
    const size_t ldx,
    const FFLAS_ELT * B,
    const size_t ldb,
    int * info)
```

**11.21.3.159 ftrtri()** [2/2]

```
template INST_OR_DECL void ftrtri (
    const FFLAS_FIELD< FFLAS_ELT > & F,
```

```

const FFLAS::FFLAS_UPLO Uplo,
const FFLAS::FFLAS_DIAG Diag,
const size_t N,
FFLAS_ELT * A,
const size_t lda,
const size_t threshold)

```

#### 11.21.3.160 trinv\_left() [2/2]

```

template INST_OR_DECL void trinv_left (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t N,
    const FFLAS_ELT * L,
    const size_t ldl,
    FFLAS_ELT * X,
    const size_t ldx)

```

#### 11.21.3.161 ftrtrm() [2/2]

```

template INST_OR_DECL void ftrtrm (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS::FFLAS_SIDE side,
    const FFLAS::FFLAS_DIAG diag,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda)

```

#### 11.21.3.162 PLUQ() [6/6]

```

template INST_OR_DECL size_t PLUQ (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS::FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda,
    size_t * P,
    size_t * Q)

```

#### 11.21.3.163 LUdivine() [4/4]

```

template INST_OR_DECL size_t LUdivine (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS::FFLAS_DIAG Diag,
    const FFLAS::FFLAS_TRANSPOSE trans,
    const size_t M,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const FFPACK_LU_TAG LuTag,
    const size_t cutoff)

```

**11.21.3.164 LUdivine\_small() [2/2]**

```
template INST_OR_DECL size_t LUdivine_small (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS::FFLAS_DIAG Diag,
    const FFLAS::FFLAS_TRANSPOSE trans,
    const size_t M,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda,
    size_t * P,
    size_t * Q,
    const FFPACK_LU_TAG LuTag)
```

**11.21.3.165 LUdivine\_gauss() [2/2]**

```
template INST_OR_DECL size_t LUdivine_gauss (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS::FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda,
    size_t * P,
    size_t * Q,
    const FFPACK_LU_TAG LuTag)
```

**11.21.3.166 RowEchelonForm() [3/3]**

```
template INST_OR_DECL size_t RowEchelonForm (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t M,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const FFPACK_LU_TAG LuTag)
```

**11.21.3.167 ReducedRowEchelonForm() [3/3]**

```
template INST_OR_DECL size_t ReducedRowEchelonForm (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t M,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const FFPACK_LU_TAG LuTag)
```

**11.21.3.168 ColumnEchelonForm()** [3/3]

```
template INST_OR_DECL size_t ColumnEchelonForm (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t M,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const FFPACK_LU_TAG LuTag)
```

**11.21.3.169 ReducedColumnEchelonForm()** [3/3]

```
template INST_OR_DECL size_t ReducedColumnEchelonForm (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t M,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const FFPACK_LU_TAG LuTag)
```

**11.21.3.170 Invert()** [3/4]

```
template INST_OR_DECL FFLAS_ELT * Invert (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t M,
    FFLAS_ELT * A,
    const size_t lda,
    int & nullity)
```

**11.21.3.171 Invert()** [4/4]

```
template INST_OR_DECL FFLAS_ELT * Invert (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t M,
    const FFLAS_ELT * A,
    const size_t lda,
    FFLAS_ELT * X,
    const size_t ldx,
    int & nullity)
```

**11.21.3.172 Invert2()** [2/2]

```
template INST_OR_DECL FFLAS_ELT * Invert2 (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t M,
    FFLAS_ELT * A,
    const size_t lda,
    FFLAS_ELT * X,
    const size_t ldx,
    int & nullity)
```

**11.21.3.173 CharPoly() [6/8]**

```
template INST_OR_DECL std::list< Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element > &
CharPoly (
    const Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > > & R,
    std::list< Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element > & charp,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda,
    FFLAS_FIELD< FFLAS_ELT >::RandIter & G,
    const FFPACK_CHARPOLY_TAG CharpTag,
    const size_t degree)
```

**11.21.3.174 CharPoly() [7/8]**

```
template INST_OR_DECL Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element & CharPoly (
    const Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > > & R,
    Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element & charp,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda,
    FFLAS_FIELD< FFLAS_ELT >::RandIter & G,
    const FFPACK_CHARPOLY_TAG CharpTag,
    const size_t degree)
```

**11.21.3.175 CharPoly() [8/8]**

```
template INST_OR_DECL Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element & CharPoly (
    const Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > > & R,
    Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element & charp,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda,
    const FFPACK_CHARPOLY_TAG CharpTag,
    const size_t degree)
```

**11.21.3.176 MinPoly() [3/4]**

```
template INST_OR_DECL std::vector< FFLAS_ELT > & MinPoly (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    std::vector< FFLAS_ELT > & minP,
    const size_t N,
    const FFLAS_ELT * A,
    const size_t lda,
    FFLAS_FIELD< FFLAS_ELT >::RandIter & G)
```

**11.21.3.177 MinPoly() [4/4]**

```
template INST_OR_DECL std::vector< FFLAS_ELT > & MinPoly (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    std::vector< FFLAS_ELT > & minP,
    const size_t N,
    const FFLAS_ELT * A,
    const size_t lda)
```



**11.21.3.178 MatVecMinPoly()** [2/2]

```
template INST_OR_DECL std::vector< FFLAS_ELT > & MatVecMinPoly (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    std::vector< FFLAS_ELT > & minP,
    const size_t N,
    const FFLAS_ELT * A,
    const size_t lda,
    const FFLAS_ELT * V,
    const size_t incv)
```

**11.21.3.179 KrylovElim()**

```
template INST_OR_DECL size_t KrylovElim (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t M,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda,
    size_t * P,
    size_t * Q,
    const size_t deg,
    size_t * iterates,
    size_t * inviterates,
    const size_t maxit,
    size_t virt)
```

**11.21.3.180 SpecRankProfile()**

```
template INST_OR_DECL size_t SpecRankProfile (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t M,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda,
    const size_t deg,
    size_t * rankProfile)
```

**11.21.3.181 Rank()** [3/3]

```
template INST_OR_DECL size_t Rank (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t M,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda)
```

**11.21.3.182 IsSingular()** [2/2]

```
template INST_OR_DECL bool IsSingular (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t M,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda)
```

**11.21.3.183 Det()** [5/6]

```
template INST_OR_DECL FFLAS_ELT & Det (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    FFLAS_ELT & det,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda,
    size_t * P,
    size_t * Q)
```

**11.21.3.184 Det()** [6/6]

```
template INST_OR_DECL FFLAS_ELT & Det (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    FFLAS_ELT & det,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda,
    const FFLAS::ParSeqHelper::Parallel< FFLAS::CuttingStrategy::Recursive, FFLAS::StrategyParameter
> & parH,
    size_t * P,
    size_t * Q)
```

**11.21.3.185 Solve()** [3/3]

```
template INST_OR_DECL FFLAS_ELT * Solve (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t M,
    FFLAS_ELT * A,
    const size_t lda,
    FFLAS_ELT * x,
    const int incx,
    const FFLAS_ELT * b,
    const int incb)
```

**11.21.3.186 solveLB()** [2/2]

```
template INST_OR_DECL void solveLB (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS::FFLAS_SIDE Side,
    const size_t M,
    const size_t N,
    const size_t R,
    FFLAS_ELT * L,
    const size_t ldl,
    const size_t * Q,
    FFLAS_ELT * B,
    const size_t ldb)
```

**11.21.3.187 solveLB2() [2/2]**

```
template INST_OR_DECL void solveLB2 (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS::FFLAS_SIDE Side,
    const size_t M,
    const size_t N,
    const size_t R,
    FFLAS_ELT * L,
    const size_t ldl,
    const size_t * Q,
    FFLAS_ELT * B,
    const size_t ldb)
```

**11.21.3.188 RandomNullSpaceVector() [3/3]**

```
template INST_OR_DECL void RandomNullSpaceVector (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS::FFLAS_SIDE Side,
    const size_t M,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda,
    FFLAS_ELT * X,
    const size_t incX)
```

**11.21.3.189 NullSpaceBasis() [2/2]**

```
template INST_OR_DECL size_t NullSpaceBasis (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS::FFLAS_SIDE Side,
    const size_t M,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda,
    FFLAS_ELT *& NS,
    size_t & ldn,
    size_t & NSdim)
```

**11.21.3.190 RowRankProfile() [3/3]**

```
template INST_OR_DECL size_t RowRankProfile (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t M,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda,
    size_t *& rkprofile,
    const FFPACK_LU_TAG LuTag)
```

**11.21.3.191 ColumnRankProfile()** [3/3]

```
template INST_OR_DECL size_t ColumnRankProfile (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t M,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda,
    size_t *& rkprofile,
    const FFPACK_LU_TAG LuTag)
```

**11.21.3.192 RowRankProfileSubmatrixIndices()** [2/2]

```
template INST_OR_DECL size_t RowRankProfileSubmatrixIndices (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t M,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda,
    size_t *& rowindices,
    size_t *& colindices,
    size_t & R)
```

**11.21.3.193 ColRankProfileSubmatrixIndices()** [2/2]

```
template INST_OR_DECL size_t ColRankProfileSubmatrixIndices (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t M,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda,
    size_t *& rowindices,
    size_t *& colindices,
    size_t & R)
```

**11.21.3.194 RowRankProfileSubmatrix()** [2/2]

```
template INST_OR_DECL size_t RowRankProfileSubmatrix (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t M,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda,
    FFLAS_ELT *& X,
    size_t & R)
```

**11.21.3.195 ColRankProfileSubmatrix()** [2/2]

```
template INST_OR_DECL size_t ColRankProfileSubmatrix (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t M,
    const size_t N,
    FFLAS_ELT * A,
    const size_t lda,
    FFLAS_ELT *& X,
    size_t & R)
```

**11.21.3.196 getTriangular< FFLAS\_FIELD< FFLAS\_ELT > >() [1/2]**

```
template INST_OR_DECL void getTriangular< FFLAS_FIELD< FFLAS_ELT > > > (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS::FFLAS_UPLO Uplo,
    const FFLAS::FFLAS_DIAG diag,
    const size_t M,
    const size_t N,
    const size_t R,
    const FFLAS_ELT * A,
    const size_t lda,
    FFLAS_ELT * T,
    const size_t ldt,
    const bool OnlyNonZeroVectors)
```

**11.21.3.197 getTriangular< FFLAS\_FIELD< FFLAS\_ELT > >() [2/2]**

```
template INST_OR_DECL void getTriangular< FFLAS_FIELD< FFLAS_ELT > > > (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS::FFLAS_UPLO Uplo,
    const FFLAS::FFLAS_DIAG diag,
    const size_t M,
    const size_t N,
    const size_t R,
    FFLAS_ELT * A,
    const size_t lda)
```

**11.21.3.198 getEchelonForm< FFLAS\_FIELD< FFLAS\_ELT > >() [1/2]**

```
template INST_OR_DECL void getEchelonForm< FFLAS_FIELD< FFLAS_ELT > > > (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS::FFLAS_UPLO Uplo,
    const FFLAS::FFLAS_DIAG diag,
    const size_t M,
    const size_t N,
    const size_t R,
    const size_t * P,
    const FFLAS_ELT * A,
    const size_t lda,
    FFLAS_ELT * T,
    const size_t ldt,
    const bool OnlyNonZeroVectors,
    const FFPACK_LU_TAG LuTag)
```

**11.21.3.199 getEchelonForm< FFLAS\_FIELD< FFLAS\_ELT > >() [2/2]**

```
template INST_OR_DECL void getEchelonForm< FFLAS_FIELD< FFLAS_ELT > > > (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS::FFLAS_UPLO Uplo,
    const FFLAS::FFLAS_DIAG diag,
    const size_t M,
    const size_t N,
    const size_t R,
    const size_t * P,
    FFLAS_ELT * A,
    const size_t lda,
    const FFPACK_LU_TAG LuTag)
```

**11.21.3.200 getEchelonTransform< FFLAS\_FIELD< FFLAS\_ELT > >()**

```
template INST_OR_DECL void getEchelonTransform< FFLAS_FIELD< FFLAS_ELT > > (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS::FFLAS_UPLO Uplo,
    const FFLAS::FFLAS_DIAG diag,
    const size_t M,
    const size_t N,
    const size_t R,
    const size_t * P,
    const size_t * Q,
    const FFLAS_ELT * A,
    const size_t lda,
    FFLAS_ELT * T,
    const size_t ldt,
    const FFPACK_LU_TAG LuTag)
```

**11.21.3.201 getReducedEchelonForm< FFLAS\_FIELD< FFLAS\_ELT > >() [1/2]**

```
template INST_OR_DECL void getReducedEchelonForm< FFLAS_FIELD< FFLAS_ELT > > (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS::FFLAS_UPLO Uplo,
    const size_t M,
    const size_t N,
    const size_t R,
    const size_t * P,
    const FFLAS_ELT * A,
    const size_t lda,
    FFLAS_ELT * T,
    const size_t ldt,
    const bool OnlyNonZeroVectors,
    const FFPACK_LU_TAG LuTag)
```

**11.21.3.202 getReducedEchelonForm< FFLAS\_FIELD< FFLAS\_ELT > >() [2/2]**

```
template INST_OR_DECL void getReducedEchelonForm< FFLAS_FIELD< FFLAS_ELT > > (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS::FFLAS_UPLO Uplo,
    const size_t M,
    const size_t N,
    const size_t R,
    const size_t * P,
    FFLAS_ELT * A,
    const size_t lda,
    const FFPACK_LU_TAG LuTag)
```

**11.21.3.203 getReducedEchelonTransform< FFLAS\_FIELD< FFLAS\_ELT > >()**

```
template INST_OR_DECL void getReducedEchelonTransform< FFLAS_FIELD< FFLAS_ELT > > (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const FFLAS::FFLAS_UPLO Uplo,
    const size_t M,
```

```

    const size_t N,
    const size_t R,
    const size_t * P,
    const size_t * Q,
    const FFLAS_ELT * A,
    const size_t lda,
    FFLAS_ELT * T,
    const size_t ldt,
    const FFPACK_LU_TAG LuTag)

```

#### 11.21.3.204 LQUPtoInverseOfFullRankMinor() [2/2]

```

template INST_OR_DECL FFLAS_ELT * LQUPtoInverseOfFullRankMinor (
    const FFLAS_FIELD< FFLAS_ELT > & F,
    const size_t rank,
    FFLAS_ELT * A_factors,
    const size_t lda,
    const size_t * QtPointer,
    FFLAS_ELT * X,
    const size_t ldx)

```

#### 11.21.3.205 fflas\_const\_cast() [3/3]

```

template<class T , class CT = const T>
T fflas_const_cast (
    CT x)

```

#### 11.21.3.206 failure()

```

Failure & failure () [inline]

```

#### 11.21.3.207 isOdd() [1/3]

```

template<class T >
bool isOdd (
    const T & a) [inline]

```

#### 11.21.3.208 isOdd() [2/3]

```

bool isOdd (
    const float & a) [inline]

```

#### 11.21.3.209 isOdd() [3/3]

```

bool isOdd (
    const double & a) [inline]

```

**11.21.3.210 NonZeroRandomMatrix()** [1/2]

```
template<class Field , class RandIter >
Field::Element_ptr NonZeroRandomMatrix (
    const Field & F,
    size_t m,
    size_t n,
    typename Field::Element_ptr A,
    size_t lda,
    RandIter & G) [inline]
```

Random non-zero Matrix.

Creates a  $m \times n$  matrix with random entries, and at least one of them is non zero.

**Parameters**

|     |       |                                                                     |
|-----|-------|---------------------------------------------------------------------|
|     | $F$   | field                                                               |
|     | $m$   | number of rows in A                                                 |
|     | $n$   | number of cols in A                                                 |
| out | $A$   | the matrix (preallocated to at least $m \times lda$ field elements) |
|     | $lda$ | leading dimension of A                                              |
|     | $G$   | a random iterator                                                   |

**Returns**

A.

**11.21.3.211 NonZeroRandomMatrix()** [2/2]

```
template<class Field , class RandIter >
Field::Element_ptr NonZeroRandomMatrix (
    const Field & F,
    size_t m,
    size_t n,
    typename Field::Element_ptr A,
    size_t lda) [inline]
```

Random non-zero Matrix.

Creates a  $m \times n$  matrix with random entries, and at least one of them is non zero.

**Parameters**

|     |       |                                                                     |
|-----|-------|---------------------------------------------------------------------|
|     | $F$   | field                                                               |
|     | $m$   | number of rows in A                                                 |
|     | $n$   | number of cols in A                                                 |
| out | $A$   | the matrix (preallocated to at least $m \times lda$ field elements) |
|     | $lda$ | leading dimension of A                                              |

**Returns**

A.



**11.21.3.212 RandomMatrix() [1/2]**

```
template<class Field , class RandIter >
Field::Element_ptr RandomMatrix (
    const Field & F,
    size_t m,
    size_t n,
    typename Field::Element_ptr A,
    size_t lda,
    RandIter & G) [inline]
```

Random Matrix.

Creates a  $m \times n$  matrix with random entries.

**Parameters**

|     |       |                                                                     |
|-----|-------|---------------------------------------------------------------------|
|     | $F$   | field                                                               |
|     | $m$   | number of rows in A                                                 |
|     | $n$   | number of cols in A                                                 |
| out | $A$   | the matrix (preallocated to at least $m \times lda$ field elements) |
|     | $lda$ | leading dimension of A                                              |
|     | $G$   | a random iterator                                                   |

**Returns**

A.

**11.21.3.213 RandomMatrix() [2/2]**

```
template<class Field >
Field::Element_ptr RandomMatrix (
    const Field & F,
    size_t m,
    size_t n,
    typename Field::Element_ptr A,
    size_t lda) [inline]
```

Random Matrix.

Creates a  $m \times n$  matrix with random entries.

**Parameters**

|     |       |                                                                     |
|-----|-------|---------------------------------------------------------------------|
|     | $F$   | field                                                               |
|     | $m$   | number of rows in A                                                 |
|     | $n$   | number of cols in A                                                 |
| out | $A$   | the matrix (preallocated to at least $m \times lda$ field elements) |
|     | $lda$ | leading dimension of A                                              |

**Returns**

A.

**11.21.3.214 RandomTriangularMatrix()** [1/2]

```
template<class Field , class RandIter >
Field::Element_ptr RandomTriangularMatrix (
    const Field & F,
    size_t m,
    size_t n,
    const FFLAS::FFLAS_UPLO UpLo,
    const FFLAS::FFLAS_DIAG Diag,
    bool nonsingular,
    typename Field::Element_ptr A,
    size_t lda,
    RandIter & G) [inline]
```

Random Triangular Matrix.

Creates a  $m \times n$  triangular matrix with random entries. The `UpLo` parameter defines whether it is upper or lower triangular.

**Parameters**

|     |             |                                                                     |
|-----|-------------|---------------------------------------------------------------------|
|     | <i>F</i>    | field                                                               |
|     | <i>m</i>    | number of rows in A                                                 |
|     | <i>n</i>    | number of cols in A                                                 |
|     | <i>UpLo</i> | whether A is upper or lower triangular                              |
| out | <i>A</i>    | the matrix (preallocated to at least $m \times lda$ field elements) |
|     | <i>lda</i>  | leading dimension of A                                              |
|     | <i>G</i>    | a random iterator                                                   |

**Returns**

A.

**11.21.3.215 RandomTriangularMatrix()** [2/2]

```
template<class Field >
Field::Element_ptr RandomTriangularMatrix (
    const Field & F,
    size_t m,
    size_t n,
    const FFLAS::FFLAS_UPLO UpLo,
    const FFLAS::FFLAS_DIAG Diag,
    bool nonsingular,
    typename Field::Element_ptr A,
    size_t lda) [inline]
```

Random Triangular Matrix.

Creates a  $m \times n$  triangular matrix with random entries. The `UpLo` parameter defines whether it is upper or lower triangular.

## Parameters

|     |        |                                                                     |
|-----|--------|---------------------------------------------------------------------|
|     | $F$    | field                                                               |
|     | $m$    | number of rows in $A$                                               |
|     | $n$    | number of cols in $A$                                               |
|     | $UpLo$ | whether $A$ is upper or lower triangular                            |
| out | $A$    | the matrix (preallocated to at least $m \times lda$ field elements) |
|     | $lda$  | leading dimension of $A$                                            |

## Returns

$A$ .

## 11.21.3.216 RandInt()

```
size_t RandInt (
    size_t a,
    size_t b) [inline]
```

## 11.21.3.217 RandomSymmetricMatrix()

```
template<class Field , class RandIter >
Field::Element_ptr RandomSymmetricMatrix (
    const Field & F,
    size_t n,
    bool nonsingular,
    typename Field::Element_ptr A,
    size_t lda,
    RandIter & G) [inline]
```

Random Symmetric Matrix.

Creates a  $m \times n$  triangular matrix with random entries. The  $UpLo$  parameter defines whether it is upper or lower triangular.

## Parameters

|     |       |                                                                     |
|-----|-------|---------------------------------------------------------------------|
|     | $F$   | field                                                               |
|     | $n$   | order of $A$                                                        |
| out | $A$   | the matrix (preallocated to at least $n \times lda$ field elements) |
|     | $lda$ | leading dimension of $A$                                            |
|     | $G$   | a random iterator                                                   |

## Returns

$A$ .

**11.21.3.218 RandomMatrixWithRank()** [1/2]

```
template<class Field , class RandIter >
Field::Element_ptr RandomMatrixWithRank (
    const Field & F,
    size_t m,
    size_t n,
    size_t r,
    typename Field::Element_ptr A,
    size_t lda,
    RandIter & G) [inline]
```

Random Matrix with prescribed rank.

Creates an  $m \times n$  matrix with random entries and rank  $r$ .

**Parameters**

|       |                                                                     |
|-------|---------------------------------------------------------------------|
| $F$   | field                                                               |
| $m$   | number of rows in A                                                 |
| $n$   | number of cols in A                                                 |
| $r$   | rank of the matrix to build                                         |
| $A$   | the matrix (preallocated to at least $m \times lda$ field elements) |
| $lda$ | leading dimension of A                                              |
| $G$   | a random iterator                                                   |

**Returns**

A.

**11.21.3.219 RandomMatrixWithRank()** [2/2]

```
template<class Field >
Field::Element_ptr RandomMatrixWithRank (
    const Field & F,
    size_t m,
    size_t n,
    size_t r,
    typename Field::Element_ptr A,
    size_t lda) [inline]
```

Random Matrix with prescribed rank.

Creates an  $m \times n$  matrix with random entries and rank  $r$ .

**Parameters**

|     |       |                                                                     |
|-----|-------|---------------------------------------------------------------------|
|     | $F$   | field                                                               |
|     | $m$   | number of rows in A                                                 |
|     | $n$   | number of cols in A                                                 |
|     | $r$   | rank of the matrix to build                                         |
| out | $A$   | the matrix (preallocated to at least $m \times lda$ field elements) |
|     | $lda$ | leading dimension of A                                              |

**Returns**

A.

**11.21.3.220 RandomIndexSubset()**

```
size_t * RandomIndexSubset (
    size_t N,
    size_t R,
    size_t * P) [inline]
```

Pick uniformly at random a sequence of  $R$  distinct elements from the set  $\{0, \dots, N - 1\}$  using Knuth's shuffle.

**Parameters**

|     |     |                                                             |
|-----|-----|-------------------------------------------------------------|
|     | $N$ | the cardinality of the sampling set                         |
|     | $R$ | the number of elements to sample                            |
| out | $P$ | the output sequence (pre-allocated to at least $R$ indices) |

**11.21.3.221 RandomPermutation()**

```
size_t * RandomPermutation (
    size_t N,
    size_t * P) [inline]
```

Pick uniformly at random a permutation of size  $N$  stored in LAPACK format using Knuth's shuffle.

**Parameters**

|     |     |                                                                |
|-----|-----|----------------------------------------------------------------|
|     | $N$ | the length of the permutation                                  |
| out | $P$ | the output permutation (pre-allocated to at least $N$ indices) |

**11.21.3.222 RandomRankProfileMatrix()**

```
void RandomRankProfileMatrix (
    size_t M,
    size_t N,
    size_t R,
    size_t * rows,
    size_t * cols) [inline]
```

Pick uniformly at random an  $R$ -subpermutation of dimension  $M \times N$  : a matrix with only  $R$  non-zeros equal to one, in a random rook placement.

**Parameters**

|     |             |                                                              |
|-----|-------------|--------------------------------------------------------------|
|     | $M$         | row dimension                                                |
|     | $N$         | column dimension                                             |
| out | <i>rows</i> | the row position of each non zero element (pre-allocated)    |
| out | <i>cols</i> | the column position of each non zero element (pre-allocated) |

**11.21.3.223 swapval()**

```
void swapval (
    size_t k,
    size_t N,
    size_t * P,
    size_t val) [inline]
```

**11.21.3.224 RandomSymmetricRankProfileMatrix()**

```
void RandomSymmetricRankProfileMatrix (
    size_t N,
    size_t R,
    size_t * rows,
    size_t * cols) [inline]
```

Pick uniformly at random a symmetric  $R$ -subpermutation of dimension  $N \times N$  : a symmetric matrix with only  $R$  non-zeros, all equal to one, in a random rook placement.

**Parameters**

|     |        |                                                              |
|-----|--------|--------------------------------------------------------------|
|     | $N$    | matrix order                                                 |
| out | $rows$ | the row position of each non zero element (pre-allocated)    |
| out | $cols$ | the column position of each non zero element (pre-allocated) |

**11.21.3.225 RandomLTQSRankProfileMatrix()**

```
void RandomLTQSRankProfileMatrix (
    size_t n,
    size_t r,
    size_t t,
    size_t * rows,
    size_t * cols) [inline]
```

**11.21.3.226 RandomMatrixWithRankandRPM()** [1/2]

```
template<class Field , class RandIter >
Field::Element_ptr RandomMatrixWithRankandRPM (
    const Field & F,
    size_t M,
    size_t N,
    size_t R,
    typename Field::Element_ptr A,
    size_t lda,
    const size_t * RRP,
    const size_t * CRP,
    RandIter & G) [inline]
```

Random Matrix with prescribed rank and rank profile matrix Creates an  $m \times n$  matrix with random entries and rank  $r$ .

## Parameters

|            |                                                                                  |
|------------|----------------------------------------------------------------------------------|
| <i>F</i>   | field                                                                            |
| <i>m</i>   | number of rows in A                                                              |
| <i>n</i>   | number of cols in A                                                              |
| <i>r</i>   | rank of the matrix to build                                                      |
| <i>A</i>   | the matrix (preallocated to at least $m \times lda$ field elements)              |
| <i>lda</i> | leading dimension of A                                                           |
| <i>RRP</i> | the R dimensional array with row positions of the rank profile matrix' pivots    |
| <i>CRP</i> | the R dimensional array with column positions of the rank profile matrix' pivots |
| <i>G</i>   | a random iterator                                                                |

## Returns

A.

## 11.21.3.227 RandomMatrixWithRankandRPM() [2/2]

```
template<class Field >
Field::Element_ptr RandomMatrixWithRankandRPM (
    const Field & F,
    size_t M,
    size_t N,
    size_t R,
    typename Field::Element_ptr A,
    size_t lda,
    const size_t * RRP,
    const size_t * CRP) [inline]
```

Random Matrix with prescribed rank and rank profile matrix Creates an  $m \times n$  matrix with random entries and rank  $r$ .

## Parameters

|            |                                                                                  |
|------------|----------------------------------------------------------------------------------|
| <i>F</i>   | field                                                                            |
| <i>m</i>   | number of rows in A                                                              |
| <i>n</i>   | number of cols in A                                                              |
| <i>r</i>   | rank of the matrix to build                                                      |
| <i>A</i>   | the matrix (preallocated to at least $m \times lda$ field elements)              |
| <i>lda</i> | leading dimension of A                                                           |
| <i>RRP</i> | the R dimensional array with row positions of the rank profile matrix' pivots    |
| <i>CRP</i> | the R dimensional array with column positions of the rank profile matrix' pivots |

## Returns

A.

**11.21.3.228 RandomSymmetricMatrixWithRankandRPM()** [1/2]

```
template<class Field , class RandIter >
Field::Element_ptr RandomSymmetricMatrixWithRankandRPM (
    const Field & F,
    size_t N,
    size_t R,
    typename Field::Element_ptr A,
    size_t lda,
    const size_t * RRP,
    const size_t * CRP,
    RandIter & G) [inline]
```

Random Symmetric Matrix with prescribed rank and rank profile matrix Creates an  $n \times n$  symmetric matrix with random entries and rank  $r$ .

**Parameters**

|       |                                                                                    |
|-------|------------------------------------------------------------------------------------|
| $F$   | field                                                                              |
| $n$   | order of $A$                                                                       |
| $r$   | rank of $A$                                                                        |
| $A$   | the matrix (preallocated to at least $n \times lda$ field elements)                |
| $lda$ | leading dimension of $A$                                                           |
| $RRP$ | the $R$ dimensional array with row positions of the rank profile matrix' pivots    |
| $CRP$ | the $R$ dimensional array with column positions of the rank profile matrix' pivots |
| $G$   | a random iterator                                                                  |

**Returns**

$A$ .

**11.21.3.229 RandomSymmetricMatrixWithRankandRPM()** [2/2]

```
template<class Field >
Field::Element_ptr RandomSymmetricMatrixWithRankandRPM (
    const Field & F,
    size_t M,
    size_t N,
    size_t R,
    typename Field::Element_ptr A,
    size_t lda,
    const size_t * RRP,
    const size_t * CRP) [inline]
```

Random Symmetric Matrix with prescribed rank and rank profile matrix Creates an  $n \times n$  symmetric matrix with random entries and rank  $r$ .

**Parameters**

|     |                                                                     |
|-----|---------------------------------------------------------------------|
| $F$ | field                                                               |
| $n$ | order of $A$                                                        |
| $r$ | rank of $A$                                                         |
| $A$ | the matrix (preallocated to at least $n \times lda$ field elements) |



|            |                                                                                         |
|------------|-----------------------------------------------------------------------------------------|
| <i>lda</i> | leading dimension of <i>A</i>                                                           |
| <i>RRP</i> | the <i>R</i> dimensional array with row positions of the rank profile matrix' pivots    |
| <i>CRP</i> | the <i>R</i> dimensional array with column positions of the rank profile matrix' pivots |

**Returns**

*A*.

**11.21.3.230 RandomMatrixWithRankandRandomRPM() [1/2]**

```
template<class Field , class RandIter >
Field::Element_ptr RandomMatrixWithRankandRandomRPM (
    const Field & F,
    size_t M,
    size_t N,
    size_t R,
    typename Field::Element_ptr A,
    size_t lda,
    RandIter & G) [inline]
```

Random Matrix with prescribed rank, with random rank profile matrix Creates an  $m \times n$  matrix with random entries, rank  $r$  and with a rank profile matrix chosen uniformly at random.

**Parameters**

|            |                                                                     |
|------------|---------------------------------------------------------------------|
| <i>F</i>   | field                                                               |
| <i>m</i>   | number of rows in <i>A</i>                                          |
| <i>n</i>   | number of cols in <i>A</i>                                          |
| <i>r</i>   | rank of the matrix to build                                         |
| <i>A</i>   | the matrix (preallocated to at least $m \times lda$ field elements) |
| <i>lda</i> | leading dimension of <i>A</i>                                       |
| <i>G</i>   | a random iterator                                                   |

**Returns**

*A*.

**11.21.3.231 RandomMatrixWithRankandRandomRPM() [2/2]**

```
template<class Field >
Field::Element_ptr RandomMatrixWithRankandRandomRPM (
    const Field & F,
    size_t M,
    size_t N,
    size_t R,
    typename Field::Element_ptr A,
    size_t lda) [inline]
```

Random Matrix with prescribed rank, with random rank profile matrix Creates an  $m \times n$  matrix with random entries, rank  $r$  and with a rank profile matrix chosen uniformly at random.

## Parameters

|       |                                                                     |
|-------|---------------------------------------------------------------------|
| $F$   | field                                                               |
| $m$   | number of rows in $A$                                               |
| $n$   | number of cols in $A$                                               |
| $r$   | rank of the matrix to build                                         |
| $A$   | the matrix (preallocated to at least $m \times lda$ field elements) |
| $lda$ | leading dimension of $A$                                            |

## Returns

$A$ .

### 11.21.3.232 RandomSymmetricMatrixWithRankandRandomRPM() [1/2]

```
template<class Field , class RandIter >
Field::Element_ptr RandomSymmetricMatrixWithRankandRandomRPM (
    const Field & F,
    size_t N,
    size_t R,
    typename Field::Element_ptr A,
    size_t lda,
    RandIter & G) [inline]
```

Random Symmetric Matrix with prescribed rank, with random rank profile matrix Creates an  $n \times n$  matrix with random entries, rank  $r$  and with a rank profile matrix chosen uniformly at random.

## Parameters

|       |                                                                     |
|-------|---------------------------------------------------------------------|
| $F$   | field                                                               |
| $n$   | order of $A$                                                        |
| $r$   | rank of $A$                                                         |
| $A$   | the matrix (preallocated to at least $n \times lda$ field elements) |
| $lda$ | leading dimension of $A$                                            |
| $G$   | a random iterator                                                   |

## Returns

$A$ .

### 11.21.3.233 RandomSymmetricMatrixWithRankandRandomRPM() [2/2]

```
template<class Field >
Field::Element_ptr RandomSymmetricMatrixWithRankandRandomRPM (
    const Field & F,
    size_t N,
    size_t R,
    typename Field::Element_ptr A,
    size_t lda) [inline]
```

Random Symmetric Matrix with prescribed rank, with random rank profile matrix Creates an  $n \times n$  matrix with random entries, rank  $r$  and with a rank profile matrix chosen uniformly at random.

## Parameters

|            |                                                                     |
|------------|---------------------------------------------------------------------|
| <i>F</i>   | field                                                               |
| <i>n</i>   | order of A                                                          |
| <i>r</i>   | rank of A                                                           |
| <i>A</i>   | the matrix (preallocated to at least $n \times lda$ field elements) |
| <i>lda</i> | leading dimension of A                                              |

## Returns

A.

**11.21.3.234 RandomMatrixWithDet()** [1/2]

```
template<class Field >
Field::Element_ptr RandomMatrixWithDet (
    const Field & F,
    size_t n,
    const typename Field::Element d,
    typename Field::Element_ptr A,
    size_t lda) [inline]
```

Random Matrix with prescribed det.

Creates a  $m \times n$  matrix with random entries and rank  $r$ .

## Parameters

|            |                                                                                     |
|------------|-------------------------------------------------------------------------------------|
| <i>F</i>   | field                                                                               |
| <i>d</i>   | the prescribed value for the determinant of A                                       |
| <i>n</i>   | number of cols in A                                                                 |
| <i>A</i>   | the matrix to be generated (preallocated to at least $n \times lda$ field elements) |
| <i>lda</i> | leading dimension of A                                                              |

## Returns

A.

**11.21.3.235 RandomMatrixWithDet()** [2/2]

```
template<class Field , class RandIter >
Field::Element_ptr RandomMatrixWithDet (
    const Field & F,
    size_t n,
    const typename Field::Element d,
    typename Field::Element_ptr A,
    size_t lda,
    RandIter & G) [inline]
```

Random Matrix with prescribed det.

Creates a  $m \times n$  matrix with random entries and rank  $r$ .

## Parameters

|       |                                                                                     |
|-------|-------------------------------------------------------------------------------------|
| $F$   | field                                                                               |
| $d$   | the prescribed value for the determinant of $A$                                     |
| $n$   | number of cols in $A$                                                               |
| $A$   | the matrix to be generated (preallocated to at least $n \times lda$ field elements) |
| $lda$ | leading dimension of $A$                                                            |

## Returns

$A$ .

**11.21.3.236 RandomLTQSMatrixWithRankandQSorder()**

```
template<class Field , class RandIter >
Field::Element_ptr RandomLTQSMatrixWithRankandQSorder (
    Field & F,
    size_t n,
    size_t r,
    size_t t,
    typename Field::Element_ptr A,
    size_t lda,
    RandIter & G) [inline]
```

**11.21.3.237 chooseField()**

```
template<typename Field >
Field * chooseField (
    Givaro::Integer q,
    uint64_t b,
    uint64_t seed)
```

**11.21.3.238 chooseField< Givaro::ZRing< int32\_t > >()**

```
template<>
Givaro::ZRing< int32_t > * chooseField< Givaro::ZRing< int32_t > > (
    Givaro::Integer q,
    uint64_t b,
    uint64_t seed)
```

**11.21.3.239 chooseField< Givaro::ZRing< int64\_t > >()**

```
template<>
Givaro::ZRing< int64_t > * chooseField< Givaro::ZRing< int64_t > > (
    Givaro::Integer q,
    uint64_t b,
    uint64_t seed)
```

**11.21.3.240 chooseField< Givaro::ZRing< float > >()**

```
template<>
Givaro::ZRing< float > * chooseField< Givaro::ZRing< float > > (
    Givaro::Integer q,
    uint64_t b,
    uint64_t seed)
```

**11.21.3.241 chooseField< Givaro::ZRing< double > >()**

```
template<>
Givaro::ZRing< double > * chooseField< Givaro::ZRing< double > > (
    Givaro::Integer q,
    uint64_t b,
    uint64_t seed)
```

**11.22 FFPACK::Protected Namespace Reference****Functions**

- template<class [Field](#) >  
size\_t [LUdivine\\_construct](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) X, const size\_t ldx, typename [Field::Element\\_ptr](#) u, const size\_t incu, size\_t \*P, bool computeX, const FFPACK\_MINPOLY\_TAG MinTag=[FpackDense](#), const size\_t kg\_mc=0, const size\_t kg\_mb=0, const size\_t kg\_j=0)
- template<class [Field](#) >  
size\_t [GaussJordan](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t colbeg, const size\_t rowbeg, const size\_t colsize, size\_t \*P, size\_t \*Q, const FFPACK::FFPACK\_LU\_TAG LuTag)  
*Gauss-Jordan algorithm computing the Reduced Row echelon form and its transform matrix.*
- template<class [Field](#) , class Polynomial >  
std::list< Polynomial > & [KellerGehrig](#) (const [Field](#) &F, std::list< Polynomial > &charp, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda)
- template<class [Field](#) , class Polynomial >  
int [KGFast](#) (const [Field](#) &F, std::list< Polynomial > &charp, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*kg\_mc, size\_t \*kg\_mb, size\_t \*kg\_j)
- template<class [Field](#) , class Polynomial >  
std::list< Polynomial > & [KGFast\\_generalized](#) (const [Field](#) &F, std::list< Polynomial > &charp, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda)
- template<class [Field](#) >  
void [fgemv\\_kgf](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, typename [Field::Element\\_ptr](#) Y, const size\_t incY, const size\_t kg\_mc, const size\_t kg\_mb, const size\_t kg\_j)
- template<class [Field](#) , class Polynomial , class RandIter >  
std::list< Polynomial > & [LUKrylov](#) (const [Field](#) &F, std::list< Polynomial > &charp, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) U, const size\_t ldu, RandIter &G)
- template<class [Field](#) , class Polynomial >  
std::list< Polynomial > & [Danilevski](#) (const [Field](#) &F, std::list< Polynomial > &charp, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda)
- template<class PolRing >  
void [RandomKrylovPrecond](#) (const PolRing &PR, std::list< typename PolRing::Element > &completedFactors, const size\_t N, typename PolRing::Domain\_t::Element\_ptr A, const size\_t lda, size\_t &Nb, typename PolRing::Domain\_t::Element\_ptr &B, size\_t &ldb, typename PolRing::Domain\_t::RandIter &g, const size\_t degree=\_\_FFLASFFPACK\_ARITHPROG\_THRESHOLD)

- `template<class PolRing >`  
`std::list< typename PolRing::Element > & ArithProg (const PolRing &PR, std::list< typename PolRing::Element > &frobeniusForm, const size_t N, typename PolRing::Domain_t::Element_ptr A, const size_t lda, const size_t degree)`
- `template<class Field , class Polynomial >`  
`std::list< Polynomial > & LUKrylov_KGFast (const Field &F, std::list< Polynomial > &charp, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr X, const size_t ldx)`
- `template<class Field , class Polynomial >`  
`Polynomial & MatVecMinPoly (const Field &F, Polynomial &minP, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr v, const size_t incv, typename Field::Element_ptr K, const size_t ldk, size_t *P)`
- `template<class Field , class Polynomial >`  
`Polynomial & Hybrid_KGF_LUK_MinPoly (const Field &F, Polynomial &minP, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr X, const size_t ldx, size_t *P, const FFPACK_MINPOLY_TAG MinTag=FFPACK::FfpackDense, const size_t kg_mc=0, const size_t kg_mb=0, const size_t kg_j=0)`
- `template<class Field >`  
`size_t updatedD (const Field &F, size_t *d, size_t k, std::vector< std::vector< typename Field::Element > > &minpt)`
- `template<class Field >`  
`size_t newD (const Field &F, size_t *d, bool &KeepOn, const size_t l, const size_t N, typename Field::Element_ptr X, const size_t *Q, std::vector< std::vector< typename Field::Element > > &minpt)`
- `template<class Field >`  
`void CompressRows (Field &F, const size_t M, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr tmp, const size_t ldtmp, const size_t *d, const size_t nb_blocs)`
- `template<class Field >`  
`void CompressRowsQK (Field &F, const size_t M, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr tmp, const size_t ldtmp, const size_t *d, const size_t deg, const size_t nb_blocs)`
- `template<class Field >`  
`void DeCompressRows (Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr tmp, const size_t ldtmp, const size_t *d, const size_t nb_blocs)`
- `template<class Field >`  
`void DeCompressRowsQK (Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr tmp, const size_t ldtmp, const size_t *d, const size_t deg, const size_t nb_blocs)`
- `template<class Field >`  
`void CompressRowsQA (Field &F, const size_t M, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr tmp, const size_t ldtmp, const size_t *d, const size_t nb_blocs)`
- `template<class Field >`  
`void DeCompressRowsQA (Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr tmp, const size_t ldtmp, const size_t *d, const size_t nb_blocs)`
- `template<class Field >`  
`size_t LUdivine_construct (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr X, const size_t ldx, typename Field::Element_ptr u, const size_t incu, size_t *P, bool computeX, const FFPACK::FFPACK_MINPOLY_TAG MinTag, const size_t kg_mc, const size_t kg_mb, const size_t kg_j)`

## 11.22.1 Function Documentation

### 11.22.1.1 LUdivine\_construct() [1/2]

```
template<class Field >
size_t LUdivine_construct (
    const Field & F,
    const FFLAS::FFLAS_DIAG Diag,
```

```

const size_t M,
const size_t N,
typename Field::ConstElement_ptr A,
const size_t lda,
typename Field::Element_ptr X,
const size_t ldx,
typename Field::Element_ptr u,
const size_t incu,
size_t * P,
bool computeX,
const FFPACK_MINPOLY_TAG MinTag = FfpackDense,
const size_t kg_mc = 0,
const size_t kg_mb = 0,
const size_t kg_j = 0)

```

### 11.22.1.2 GaussJordan()

```

template<class Field >
size_t GaussJordan (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    const size_t colbeg,
    const size_t rowbeg,
    const size_t colsize,
    size_t * P,
    size_t * Q,
    const FFPACK::FFPACK_LU_TAG LuTag) [inline]

```

Gauss-Jordan algorithm computing the Reduced Row echelon form and its transform matrix.

- Bibliography**
- Algorithm 2.8 of A. Storjohann Thesis 2000,
  - Algorithm 11 of Jeannerod C-P., Pernet, C. and Storjohann, A. *Rank-profile revealing Gaussian elimination and the CUP matrix decomposition*, J. of Symbolic Comp., 2013

#### Parameters

|                |              |                                                                                                          |
|----------------|--------------|----------------------------------------------------------------------------------------------------------|
|                | <i>M</i>     | row dimension of A                                                                                       |
|                | <i>N</i>     | column dimension of A                                                                                    |
| <i>in, out</i> | <i>A</i>     | an m x n matrix                                                                                          |
|                | <i>lda</i>   | leading dimension of A                                                                                   |
|                | <i>P</i>     | row permutation                                                                                          |
|                | <i>Q</i>     | column permutation                                                                                       |
|                | <i>LuTag</i> | set the base case to a Tile (FfpackGaussJordanTile) or Slab (FfpackGaussJordanSlab) recursive RedEchelon |

where the transformation matrix is stored at the pivot column position

**11.22.1.3 KellerGehrig()**

```
template<class Field , class Polynomial >
std::list< Polynomial > & KellerGehrig (
    const Field & F,
    std::list< Polynomial > & charp,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t lda)
```

**11.22.1.4 KGFast()**

```
template<class Field , class Polynomial >
int KGFast (
    const Field & F,
    std::list< Polynomial > & charp,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * kg_mc,
    size_t * kg_mb,
    size_t * kg_j)
```

**11.22.1.5 KGFast\_generalized()**

```
template<class Field , class Polynomial >
std::list< Polynomial > & KGFast_generalized (
    const Field & F,
    std::list< Polynomial > & charp,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda)
```

**11.22.1.6 fgemv\_kgf()**

```
template<class Field >
void fgemv_kgf (
    const Field & F,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr X,
    const size_t incX,
    typename Field::Element_ptr Y,
    const size_t incY,
    const size_t kg_mc,
    const size_t kg_mb,
    const size_t kg_j)
```

**11.22.1.7 LUKrylov()**

```
template<class Field , class Polynomial , class RandIter >
std::list< Polynomial > & LUKrylov (
    const Field & F,
    std::list< Polynomial > & charp,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr U,
```



```
const size_t ldu,
RandIter & G)
```

### 11.22.1.8 Danilevski()

```
template<class Field , class Polynomial >
std::list< Polynomial > & Danilevski (
    const Field & F,
    std::list< Polynomial > & charp,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda)
```

### 11.22.1.9 RandomKrylovPrecond()

```
template<class PolRing >
void RandomKrylovPrecond (
    const PolRing & PR,
    std::list< typename PolRing::Element > & completedFactors,
    const size_t N,
    typename PolRing::Domain_t::Element_ptr A,
    const size_t lda,
    size_t & Nb,
    typename PolRing::Domain_t::Element_ptr & B,
    size_t & ldb,
    typename PolRing::Domain_t::RandIter & g,
    const size_t degree = __FFLASFFPACK_ARITHPROG_THRESHOLD) [inline]
```

**Todo** swap to save space ??

**Todo**

**Todo** don't assing K2 c\*noc x N but only mas (c,noc) x N and store each one after the other

**Todo** swap to save space ??

**Todo**

**Todo** don't assing K2 c\*noc x N but only mas (c,noc) x N and store each one after the other

### 11.22.1.10 ArithProg()

```
template<class PolRing >
std::list< typename PolRing::Element > & ArithProg (
    const PolRing & PR,
    std::list< typename PolRing::Element > & frobeniusForm,
    const size_t N,
    typename PolRing::Domain_t::Element_ptr A,
    const size_t lda,
    const size_t degree) [inline]
```

### 11.22.1.11 LUKrylov\_KGFast()

```
template<class Field , class Polynomial >
std::list< Polynomial > & LUKrylov_KGFast (
    const Field & F,
    std::list< Polynomial > & charp,
    const size_t N,
```

```

typename Field::Element_ptr A,
const size_t lda,
typename Field::Element_ptr X,
const size_t ldX)

```

#### 11.22.1.12 MatVecMinPoly()

```

template<class Field , class Polynomial >
Polynomial & MatVecMinPoly (
    const Field & F,
    Polynomial & minP,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::Element_ptr v,
    const size_t incv,
    typename Field::Element_ptr K,
    const size_t ldK,
    size_t * P) [inline]

```

#### 11.22.1.13 Hybrid\_KGF\_LUK\_MinPoly()

```

template<class Field , class Polynomial >
Polynomial & Hybrid_KGF_LUK_MinPoly (
    const Field & F,
    Polynomial & minP,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::Element_ptr X,
    const size_t ldX,
    size_t * P,
    const FFPACK_MINPOLY_TAG MinTag = FFPACK::FfpackDense,
    const size_t kg_mc = 0,
    const size_t kg_mb = 0,
    const size_t kg_j = 0)

```

#### 11.22.1.14 updateD()

```

template<class Field >
size_t updateD (
    const Field & F,
    size_t * d,
    size_t k,
    std::vector< std::vector< typename Field::Element > > & minpt)

```

#### 11.22.1.15 newD()

```

template<class Field >
size_t newD (
    const Field & F,
    size_t * d,
    bool & KeepOn,
    const size_t l,
    const size_t N,
    typename Field::Element_ptr X,
    const size_t * Q,
    std::vector< std::vector< typename Field::Element > > & minpt)

```

**11.22.1.16 CompressRows()**

```
template<class Field >
void CompressRows (
    Field & F,
    const size_t M,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr tmp,
    const size_t ldtmp,
    const size_t * d,
    const size_t nb_blocs) [inline]
```

**11.22.1.17 CompressRowsQK()**

```
template<class Field >
void CompressRowsQK (
    Field & F,
    const size_t M,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr tmp,
    const size_t ldtmp,
    const size_t * d,
    const size_t deg,
    const size_t nb_blocs) [inline]
```

**11.22.1.18 DeCompressRows()**

```
template<class Field >
void DeCompressRows (
    Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr tmp,
    const size_t ldtmp,
    const size_t * d,
    const size_t nb_blocs) [inline]
```

**11.22.1.19 DeCompressRowsQK()**

```
template<class Field >
void DeCompressRowsQK (
    Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr tmp,
    const size_t ldtmp,
    const size_t * d,
    const size_t deg,
    const size_t nb_blocs) [inline]
```

**11.22.1.20 CompressRowsQA()**

```
template<class Field >
void CompressRowsQA (
```

```

    Field & F,
    const size_t M,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr tmp,
    const size_t ldtmp,
    const size_t * d,
    const size_t nb_blocs) [inline]

```

#### 11.22.1.21 DeCompressRowsQA()

```

template<class Field >
void DeCompressRowsQA (
    Field & F,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    typename Field::Element_ptr tmp,
    const size_t ldtmp,
    const size_t * d,
    const size_t nb_blocs) [inline]

```

#### 11.22.1.22 LUdivine\_construct() [2/2]

```

template<class Field >
size_t LUdivine_construct (
    const Field & F,
    const FFLAS::FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::Element_ptr X,
    const size_t ldx,
    typename Field::Element_ptr u,
    const size_t incu,
    size_t * P,
    bool computeX,
    const FFPACK::FFPACK_MINPOLY_TAG MinTag,
    const size_t kg_mc,
    const size_t kg_mb,
    const size_t kg_j)

```

## 11.23 Givaro Namespace Reference

### Data Structures

- class [ModularBalanced](#)
- class [Montgomery](#)

## 11.24 MKL\_CONFIG Namespace Reference

## 11.25 Reclnt Namespace Reference

### Data Structures

- class [rint](#)

- class [ruint](#)



# Chapter 12

## Data Structure Documentation

### 12.1 AlgoChooser< ModeT, ParSeq > Struct Template Reference

#### Public Types

- typedef [MMHelperAlgo::Winograd value](#)

#### 12.1.1 Member Typedef Documentation

##### 12.1.1.1 value

```
template<class ModeT , class ParSeq >
MMHelperAlgo::Winograd value
```

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

### 12.2 AlgoChooser< ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq > Struct Template Reference

#### Public Types

- typedef [MMHelperAlgo::Classic value](#)

#### 12.2.1 Member Typedef Documentation

##### 12.2.1.1 value

```
template<class ParSeq >
MMHelperAlgo::Classic value
```

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

### 12.3 ALL< v > Struct Template Reference

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.4 ALL< false, v... > Struct Template Reference

### Static Public Attributes

- static constexpr bool [value](#) = false

### 12.4.1 Field Documentation

#### 12.4.1.1 value

```
template<bool... v>
bool value = false [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.5 ALL< true, v... > Struct Template Reference

### Static Public Attributes

- static constexpr bool [value](#) = ALL<v...>::value

### 12.5.1 Field Documentation

#### 12.5.1.1 value

```
template<bool... v>
bool value = ALL<v...>::value [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.6 ALL<> Struct Reference

### Static Public Attributes

- static constexpr bool [value](#) = true

### 12.6.1 Field Documentation

#### 12.6.1.1 value

```
bool value = true [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.7 ArbitraryPrecIntTag Struct Reference

Arbitrary precision integers: GMP.

```
#include <field-traits.h>
```

### 12.7.1 Detailed Description

Arbitrary precision integers: GMP.

The documentation for this struct was generated from the following file:

- [field-traits.h](#)



## 12.8 AreEqual< X, Y > Class Template Reference

```
#include <fflas_enum.h>
```

### Static Public Attributes

- static const bool [value](#) = false

### 12.8.1 Field Documentation

#### 12.8.1.1 value

```
template<class X , class Y >
const bool value = false [static]
```

The documentation for this class was generated from the following file:

- [fflas\\_enum.h](#)

## 12.9 AreEqual< X, X > Class Template Reference

```
#include <fflas_enum.h>
```

### Static Public Attributes

- static const bool [value](#) = true

### 12.9.1 Field Documentation

#### 12.9.1.1 value

```
template<class X >
const bool value = true [static]
```

The documentation for this class was generated from the following file:

- [fflas\\_enum.h](#)

## 12.10 Argument Struct Reference

```
#include <args-parser.h>
```

### Data Fields

- char [c](#)
- const char \* [example](#)
- const char \* [helpString](#)
- [ArgumentType](#) type
- void \* [data](#)

### 12.10.1 Field Documentation

#### 12.10.1.1 c

```
char c
```

#### 12.10.1.2 example

```
const char* example
```

### 12.10.1.3 helpString

```
const char* helpString
```

### 12.10.1.4 type

```
ArgumentType type
```

### 12.10.1.5 data

```
void* data
```

The documentation for this struct was generated from the following file:

- [args-parser.h](#)

## 12.11 associatedDelayedField< Field > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [Field](#) [field](#)
- typedef [Field](#) & [type](#)

### 12.11.1 Member Typedef Documentation

#### 12.11.1.1 field

```
template<class Field >
Field field
```

#### 12.11.1.2 type

```
template<class Field >
Field& type
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.12 associatedDelayedField< const FFPACK::RNSIntegerMod< RNS > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [FFPACK::RNSInteger](#)< [RNS](#) > [field](#)
- typedef [FFPACK::RNSInteger](#)< [RNS](#) > [type](#)

### 12.12.1 Member Typedef Documentation

#### 12.12.1.1 field

```
template<typename RNS >
FFPACK::RNSInteger<RNS> field
```

### 12.12.1.2 type

```
template<typename RNS >
FFPACK::RNSInteger<RNS> type
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.13 associatedDelayedField< const Givaro::Modular< T, X > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- typedef Givaro::ZRing< T > [field](#)
- typedef Givaro::ZRing< T > [type](#)

### 12.13.1 Member Typedef Documentation

#### 12.13.1.1 field

```
template<typename T , typename X >
Givaro::ZRing<T> field
```

#### 12.13.1.2 type

```
template<typename T , typename X >
Givaro::ZRing<T> type
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.14 associatedDelayedField< const Givaro::ModularBalanced< T > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- typedef Givaro::ZRing< T > [field](#)
- typedef Givaro::ZRing< T > [type](#)

### 12.14.1 Member Typedef Documentation

#### 12.14.1.1 field

```
template<typename T >
Givaro::ZRing<T> field
```

#### 12.14.1.2 type

```
template<typename T >
Givaro::ZRing<T> type
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.15 associatedDelayedField< const Givaro::ZRing< T > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- typedef Givaro::ZRing< T > [field](#)
- typedef Givaro::ZRing< T > [type](#)

### 12.15.1 Member Typedef Documentation

#### 12.15.1.1 field

```
template<typename T >
Givaro::ZRing<T> field
```

#### 12.15.1.2 type

```
template<typename T >
Givaro::ZRing<T> type
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.16 Auto Struct Reference

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

## 12.17 Bench< Elt > Class Template Reference

### Public Types

- using [Field](#) = Modular<Elt>
- using [Elt\\_ptr](#) = typename [Field::Element\\_ptr](#)
- using [Residu](#) = typename [Field::Residu\\_t](#)
- template<bool B, class T = void>
 using [enable\\_if\\_t](#) = typename std::enable\_if<B, T>::type
- template<typename [Simd](#) >
 using [is\\_same\\_element](#) = typename [Simd::template is\\_same\\_element<\[Field\]\(#\)>](#)
- template<typename E >
 using [enable\\_if\\_no\\_simd\\_t](#) = [enable\\_if\\_t](#)<[Simd](#)<E>::vect\_size == 1>
- template<typename E >
 using [enable\\_if\\_simd128\\_t](#) = [enable\\_if\\_t](#)<sizeof(E)\*[Simd](#)<E>::vect\_size == 16>
- template<typename E >
 using [enable\\_if\\_simd256\\_t](#) = [enable\\_if\\_t](#)<sizeof(E)\*[Simd](#)<E>::vect\_size == 32>
- template<typename E >
 using [enable\\_if\\_simd512\\_t](#) = [enable\\_if\\_t](#)<sizeof(E)\*[Simd](#)<E>::vect\_size == 64>

### Public Member Functions

- [Bench](#) (size\_t m, size\_t n, size\_t iters, bool inplace)
- template<typename [Simd](#) = NoSimd<Elt>, [enable\\_if\\_t](#)< [is\\_same\\_element](#)< [Simd](#)>::value > \* = nullptr>
 void [doBenchs](#) ()
- template<typename \_E = Elt, [enable\\_if\\_t](#)< is\_same< \_E, Elt >::value > \* = nullptr, [enable\\_if\\_no\\_simd\\_t](#)< \_E > \* = nullptr>
 void [run](#) (bool allsimd)

## Static Public Member Functions

- `template<typename _E = Elt, enable\_if\_t<lis_same<_E, Givaro::Integer>::value> * = nullptr>`  
static [Residu cardinality](#) ()
- `template<typename _E = Elt, enable\_if\_t<is_same<_E, Givaro::Integer>::value> * = nullptr>`  
static [Residu cardinality](#) ()

## Protected Attributes

- [Field F](#)
- `const size_t m`
- `const size_t n`
- `const size_t iters`
- `const bool inplace`

## 12.17.1 Member Typedef Documentation

### 12.17.1.1 Field

```
template<typename Elt >
using Field = Modular<Elt>
```

### 12.17.1.2 Elt\_ptr

```
template<typename Elt >
using Elt\_ptr = typename Field::Element\_ptr
```

### 12.17.1.3 Residu

```
template<typename Elt >
using Residu = typename Field::Residu\_t
```

### 12.17.1.4 enable\_if\_t

```
template<typename Elt >
template<bool B, class T = void>
using enable\_if\_t = typename std::enable_if<B, T>::type
```

### 12.17.1.5 is\_same\_element

```
template<typename Elt >
template<typename Simd >
using is\_same\_element = typename Simd::template is\_same\_element<Field>
```

### 12.17.1.6 enable\_if\_no\_simd\_t

```
template<typename Elt >
template<typename E >
using enable\_if\_no\_simd\_t = enable\_if\_t<Simd<E>::vect_size == 1>
```

### 12.17.1.7 enable\_if\_simd128\_t

```
template<typename Elt >
template<typename E >
using enable\_if\_simd128\_t = enable\_if\_t<sizeof(E)*Simd<E>::vect_size == 16>
```

### 12.17.1.8 enable\_if\_simd256\_t

```
template<typename Elt >
template<typename E >
using enable\_if\_simd256\_t = enable\_if\_t<sizeof(E)*Simd<E>::vect_size == 32>
```

### 12.17.1.9 enable\_if\_simd512\_t

```
template<typename Elt >
template<typename E >
using enable_if_simd512_t = enable_if_t<sizeof(E)*Simd<E>::vect_size == 64>
```

## 12.17.2 Constructor & Destructor Documentation

### 12.17.2.1 Bench()

```
template<typename Elt >
Bench (
    size_t m,
    size_t n,
    size_t iters,
    bool inplace) [inline]
```

## 12.17.3 Member Function Documentation

### 12.17.3.1 cardinality() [1/2]

```
template<typename Elt >
template<typename _E = Elt, enable_if_t<!is_same< _E, Givaro::Integer >::value > * = nullptr>
static Residu cardinality () [inline], [static]
```

### 12.17.3.2 cardinality() [2/2]

```
template<typename Elt >
template<typename _E = Elt, enable_if_t< is_same< _E, Givaro::Integer >::value > * = nullptr>
static Residu cardinality () [inline], [static]
```

### 12.17.3.3 doBenchs()

```
template<typename Elt >
template<typename Simd = NoSimd<Elt>, enable_if_t< is_same_element< Simd >::value > * =
nullptr>
void doBenchs () [inline]
```

### 12.17.3.4 run()

```
template<typename Elt >
template<typename _E = Elt, enable_if_t< is_same< _E, Elt >::value > * = nullptr, enable_if_no_simd_t<
_E > * = nullptr>
void run (
    bool allsimd) [inline]
```

## 12.17.4 Field Documentation

### 12.17.4.1 F

```
template<typename Elt >
Field F [protected]
```

### 12.17.4.2 m

```
template<typename Elt >
const size_t m [protected]
```

**12.17.4.3 n**

```
template<typename Elt >
const size_t n [protected]
```

**12.17.4.4 iters**

```
template<typename Elt >
const size_t iters [protected]
```

**12.17.4.5 inplace**

```
template<typename Elt >
const bool inplace [protected]
```

The documentation for this class was generated from the following file:

- [benchmark-storage-transpose.C](#)

**12.18 Bini Struct Reference**

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

**12.19 Block Struct Reference**

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

**12.20 BlockTransposeSIMD< Field, Simd, > Struct Template Reference**

```
#include <fflas_transpose.h>
```

**Public Member Functions**

- template<size\_t s = Simd::vect\_size, IsSimdSize< s, 1 > \* = nullptr>  
void [transpose](#) (const [Field](#) &F, ConstElement\_ptr A, size\_t lda, Element\_ptr B, size\_t ldb) const
- template<size\_t s = Simd::vect\_size, IsSimdSize< s, 2 > \* = nullptr>  
void [transpose](#) (const [Field](#) &F, ConstElement\_ptr A, size\_t lda, Element\_ptr B, size\_t ldb) const
- template<size\_t s = Simd::vect\_size, IsSimdSize< s, 4 > \* = nullptr>  
void [transpose](#) (const [Field](#) &F, ConstElement\_ptr A, size\_t lda, Element\_ptr B, size\_t ldb) const
- template<size\_t s = Simd::vect\_size, IsSimdSize< s, 8 > \* = nullptr>  
void [transpose](#) (const [Field](#) &F, ConstElement\_ptr A, size\_t lda, Element\_ptr B, size\_t ldb) const
- template<size\_t s = Simd::vect\_size, IsSimdSize< s, 16 > \* = nullptr>  
void [transpose](#) (const [Field](#) &F, ConstElement\_ptr A, size\_t lda, Element\_ptr B, size\_t ldb) const

**Static Public Member Functions**

- static constexpr size\_t [size](#) ()
- static const std::string [info](#) ()

**12.20.1 Member Function Documentation****12.20.1.1 size()**

```
template<typename Field , typename Simd , typename std::enable_if< Simd::template is_same_↔
element< Field >::value >::type * = nullptr>
static constexpr size_t size () [inline], [static], [constexpr]
```

**12.20.1.2 info()**

```
template<typename Field , typename Simd , typename std::enable_if< Simd::template is_same_↵
element< Field >::value >::type * = nullptr>
static const std::string info () [inline], [static]
```

**12.20.1.3 transpose() [1/5]**

```
template<typename Field , typename Simd , typename std::enable_if< Simd::template is_same_↵
element< Field >::value >::type * = nullptr>
template<size_t _s = Simd::vect_size, IsSimdSize< _s, 1 > * = nullptr>
void transpose (
    const Field & F,
    ConstElement_ptr A,
    size_t lda,
    Element_ptr B,
    size_t ldb) const [inline]
```

**12.20.1.4 transpose() [2/5]**

```
template<typename Field , typename Simd , typename std::enable_if< Simd::template is_same_↵
element< Field >::value >::type * = nullptr>
template<size_t _s = Simd::vect_size, IsSimdSize< _s, 2 > * = nullptr>
void transpose (
    const Field & F,
    ConstElement_ptr A,
    size_t lda,
    Element_ptr B,
    size_t ldb) const [inline]
```

**12.20.1.5 transpose() [3/5]**

```
template<typename Field , typename Simd , typename std::enable_if< Simd::template is_same_↵
element< Field >::value >::type * = nullptr>
template<size_t _s = Simd::vect_size, IsSimdSize< _s, 4 > * = nullptr>
void transpose (
    const Field & F,
    ConstElement_ptr A,
    size_t lda,
    Element_ptr B,
    size_t ldb) const [inline]
```

**12.20.1.6 transpose() [4/5]**

```
template<typename Field , typename Simd , typename std::enable_if< Simd::template is_same_↵
element< Field >::value >::type * = nullptr>
template<size_t _s = Simd::vect_size, IsSimdSize< _s, 8 > * = nullptr>
void transpose (
    const Field & F,
    ConstElement_ptr A,
    size_t lda,
    Element_ptr B,
    size_t ldb) const [inline]
```

**12.20.1.7 transpose() [5/5]**

```
template<typename Field , typename Simd , typename std::enable_if< Simd::template is_same_↵
element< Field >::value >::type * = nullptr>
template<size_t _s = Simd::vect_size, IsSimdSize< _s, 16 > * = nullptr>
```



```
void transpose (
    const Field & F,
    ConstElement_ptr A,
    size_t lda,
    Element_ptr B,
    size_t ldb) const [inline]
```

The documentation for this struct was generated from the following file:

- [fflas\\_transpose.h](#)

## 12.21 callLUdivine\_small< Element > Class Template Reference

### Public Member Functions

- `template<class Field >`  
`size_t operator() (const Field &F, const FFLAS::FFLAS_DIAG Diag, const FFLAS::FFLAS_TRANSPOSE trans, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, const FFPACK::FFPACK_LU_TAG LuTag)`

### 12.21.1 Member Function Documentation

#### 12.21.1.1 operator>()

```
template<class Element >
template<class Field >
size_t operator() (
    const Field & F,
    const FFLAS::FFLAS_DIAG Diag,
    const FFLAS::FFLAS_TRANSPOSE trans,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t * Q,
    const FFPACK::FFPACK_LU_TAG LuTag) [inline]
```

The documentation for this class was generated from the following file:

- [ffpack\\_ludivine.inl](#)

## 12.22 callLUdivine\_small< double > Class Reference

### Public Member Functions

- `template<class Field >`  
`size_t operator() (const Field &F, const FFLAS::FFLAS_DIAG Diag, const FFLAS::FFLAS_TRANSPOSE trans, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, const FFPACK::FFPACK_LU_TAG LuTag)`

### 12.22.1 Member Function Documentation

#### 12.22.1.1 operator>()

```
template<class Field >
size_t operator() (
    const Field & F,
    const FFLAS::FFLAS_DIAG Diag,
    const FFLAS::FFLAS_TRANSPOSE trans,
    const size_t M,
```

```

const size_t N,
typename Field::Element_ptr A,
const size_t lda,
size_t * P,
size_t * Q,
const FFPACK::FFPACK_LU_TAG LuTag) [inline]

```

The documentation for this class was generated from the following file:

- [ffpack\\_ludivine.inl](#)

## 12.23 callLUdivine\_small< float > Class Reference

### Public Member Functions

- `template<class Field >`  
`size_t operator() (const Field &F, const FFLAS::FFLAS_DIAG Diag, const FFLAS::FFLAS_TRANSPOSE trans, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, const FFPACK::FFPACK_LU_TAG LuTag)`

### 12.23.1 Member Function Documentation

#### 12.23.1.1 operator>()

```

template<class Field >
size_t operator() (
    const Field & F,
    const FFLAS::FFLAS_DIAG Diag,
    const FFLAS::FFLAS_TRANSPOSE trans,
    const size_t M,
    const size_t N,
    typename Field::Element_ptr A,
    const size_t lda,
    size_t * P,
    size_t * Q,
    const FFPACK::FFPACK_LU_TAG LuTag) [inline]

```

The documentation for this class was generated from the following file:

- [ffpack\\_ludivine.inl](#)

## 12.24 CharpolyFailed Class Reference

```
#include <ffpack.h>
```

The documentation for this class was generated from the following file:

- [ffpack.h](#)

## 12.25 Checker\_Empty< Field > Struct Template Reference

```
#include <checker_empty.h>
```

### Public Member Functions

- `template<typename... Params>`  
`Checker_Empty (Params... parameters)`
- `template<typename... Params>`  
`bool check (Params... parameters)`

## 12.25.1 Constructor & Destructor Documentation

### 12.25.1.1 Checker\_Empty()

```
template<class Field >
template<typename... Params>
Checker_Empty (
    Params... parameters) [inline]
```

## 12.25.2 Member Function Documentation

### 12.25.2.1 check()

```
template<class Field >
template<typename... Params>
bool check (
    Params... parameters) [inline]
```

The documentation for this struct was generated from the following file:

- [checker\\_empty.h](#)

## 12.26 CheckerImplem\_charpoly< Field, Polynomial > Class Template Reference

### Public Member Functions

- [CheckerImplem\\_charpoly](#) (const [Field](#) &F\_, const size\_t n\_, typename [Field::ConstElement\\_ptr](#) A, size\_t lda\_)
- [CheckerImplem\\_charpoly](#) (typename [Field::RandIter](#) &G, const size\_t n\_, typename [Field::ConstElement\\_ptr](#) A, size\_t lda\_)
- [~CheckerImplem\\_charpoly](#) ()
- bool [check](#) (Polynomial &g)

## 12.26.1 Constructor & Destructor Documentation

### 12.26.1.1 CheckerImplem\_charpoly() [1/2]

```
template<class Field , class Polynomial >
CheckerImplem_charpoly (
    const Field & F_,
    const size_t n_,
    typename Field::ConstElement_ptr A,
    size_t lda_) [inline]
```

### 12.26.1.2 CheckerImplem\_charpoly() [2/2]

```
template<class Field , class Polynomial >
CheckerImplem_charpoly (
    typename Field::RandIter & G,
    const size_t n_,
    typename Field::ConstElement_ptr A,
    size_t lda_) [inline]
```

### 12.26.1.3 ~CheckerImplem\_charpoly()

```
template<class Field , class Polynomial >
~CheckerImplem_charpoly () [inline]
```

## 12.26.2 Member Function Documentation

### 12.26.2.1 check()

```
template<class Field , class Polynomial >
bool check (
    Polynomial & g) [inline]
```

The documentation for this class was generated from the following files:

- [checker\\_charpoly.inl](#)
- [checkers\\_ffpack.h](#)

## 12.27 CheckerImplem\_charpoly< Givaro::ZRing< Givaro::Integer >, Polynomial > Class Template Reference

### Public Types

- typedef Givaro::ZRing< Givaro::Integer > [Ring](#)

### Public Member Functions

- [CheckerImplem\\_charpoly](#) (const [Ring](#) &F\_, const size\_t n\_, typename Ring::ConstElement\_ptr A, size\_t lda\_)
- [CheckerImplem\\_charpoly](#) (typename Ring::RandIter &G, const size\_t n\_, typename Ring::ConstElement\_ptr A, size\_t lda\_)
- [~CheckerImplem\\_charpoly](#) ()
- bool [check](#) (Polynomial &g)

## 12.27.1 Member Typedef Documentation

### 12.27.1.1 Ring

```
template<class Polynomial >
Givaro::ZRing<Givaro::Integer> Ring
```

## 12.27.2 Constructor & Destructor Documentation

### 12.27.2.1 CheckerImplem\_charpoly() [1/2]

```
template<class Polynomial >
CheckerImplem\_charpoly (
    const Ring & F_,
    const size_t n_,
    typename Ring::ConstElement_ptr A,
    size_t lda_) [inline]
```

### 12.27.2.2 CheckerImplem\_charpoly() [2/2]

```
template<class Polynomial >
CheckerImplem\_charpoly (
    typename Ring::RandIter & G,
    const size_t n_,
    typename Ring::ConstElement_ptr A,
    size_t lda_) [inline]
```

### 12.27.2.3 ~CheckerImplem\_charpoly()

```
template<class Polynomial >
~CheckerImplem\_charpoly () [inline]
```

## 12.27.3 Member Function Documentation

### 12.27.3.1 check()

```
template<class Polynomial >
bool check (
    Polynomial & g) [inline]
```

The documentation for this class was generated from the following file:

- [checker\\_charpoly.inl](#)

## 12.28 CheckerImplem\_Det< Field > Class Template Reference

### Public Member Functions

- [CheckerImplem\\_Det](#) (const [Field](#) &F\_, size\_t n\_, typename [Field::ConstElement\\_ptr](#) A, size\_t lda)
- [CheckerImplem\\_Det](#) (typename [Field::RandIter](#) &G, size\_t n\_, typename [Field::ConstElement\\_ptr](#) A, size\_t lda)
- [~CheckerImplem\\_Det](#) ()
- bool [check](#) (const typename [Field::Element](#) &det, typename [Field::ConstElement\\_ptr](#) LU, size\_t lda, size\_t \*P, size\_t \*Q) const  
*check if the Det factorization is correct.*

## 12.28.1 Constructor & Destructor Documentation

### 12.28.1.1 CheckerImplem\_Det() [1/2]

```
template<class Field >
CheckerImplem_Det (
    const Field & F_,
    size_t n_,
    typename Field::ConstElement_ptr A,
    size_t lda) [inline]
```

### 12.28.1.2 CheckerImplem\_Det() [2/2]

```
template<class Field >
CheckerImplem_Det (
    typename Field::RandIter & G,
    size_t n_,
    typename Field::ConstElement_ptr A,
    size_t lda) [inline]
```

### 12.28.1.3 ~CheckerImplem\_Det()

```
template<class Field >
~CheckerImplem_Det () [inline]
```

## 12.28.2 Member Function Documentation

### 12.28.2.1 check()

```
template<class Field >
bool check (
    const typename Field::Element & det,
    typename Field::ConstElement_ptr LU,
    size_t lda,
    size_t * P,
    size_t * Q) const [inline]
```

check if the Det factorization is correct.

Needs matrix in LU form

## Parameters

|                   |             |
|-------------------|-------------|
| <i>LU,storage</i> | for L and U |
| <i>det</i>        |             |
| <i>P</i>          |             |
| <i>Q</i>          |             |

The documentation for this class was generated from the following files:

- [checker\\_det.ini](#)
- [checkers\\_ffpack.h](#)

## 12.29 CheckerImplem\_fgemm< Field > Class Template Reference

### Public Member Functions

- [CheckerImplem\\_fgemm](#) (const [Field](#) &F\_, const size\_t m\_, const size\_t n\_, const size\_t k\_, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc\_)
- [CheckerImplem\\_fgemm](#) (typename [Field::RandIter](#) &G, const size\_t m\_, const size\_t n\_, const size\_t k\_, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc\_)
- [~CheckerImplem\\_fgemm](#) ()
- bool [check](#) (const [FFLAS::FFLAS\\_TRANSPOSE](#) ta, const [FFLAS::FFLAS\\_TRANSPOSE](#) tb, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::ConstElement\\_ptr](#) C)

### 12.29.1 Constructor & Destructor Documentation

#### 12.29.1.1 CheckerImplem\_fgemm() [1/2]

```
template<class Field >
CheckerImplem_fgemm (
    const Field & F_,
    const size_t m_,
    const size_t n_,
    const size_t k_,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc_) [inline]
```

#### 12.29.1.2 CheckerImplem\_fgemm() [2/2]

```
template<class Field >
CheckerImplem_fgemm (
    typename Field::RandIter & G,
    const size_t m_,
    const size_t n_,
    const size_t k_,
    const typename Field::Element beta,
    typename Field::Element_ptr C,
    const size_t ldc_) [inline]
```

#### 12.29.1.3 ~CheckerImplem\_fgemm()

```
template<class Field >
~CheckerImplem_fgemm () [inline]
```

## 12.29.2 Member Function Documentation

### 12.29.2.1 check()

```
template<class Field >
bool check (
    const FFLAS::FFLAS_TRANSPOSE ta,
    const FFLAS::FFLAS_TRANSPOSE tb,
    const typename Field::Element alpha,
    typename Field::ConstElement_ptr A,
    const size_t lda,
    typename Field::ConstElement_ptr B,
    const size_t ldb,
    typename Field::ConstElement_ptr C) [inline]
```

The documentation for this class was generated from the following files:

- [checker\\_fgemv.inl](#)
- [checkers\\_fflas.h](#)

## 12.30 CheckerImplem\_ftrsm< Field > Class Template Reference

### Public Member Functions

- [CheckerImplem\\_ftrsm](#) (const [Field](#) &F\_, const size\_t m, const size\_t n, const typename [Field::Element](#) alpha, const typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb)
- [CheckerImplem\\_ftrsm](#) (typename [Field::RandIter](#) &G, const size\_t m, const size\_t n, const typename [Field::Element](#) alpha, const typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb)
- [~CheckerImplem\\_ftrsm](#) ()
- bool [check](#) (const [FFLAS::FFLAS\\_SIDE](#) side, const [FFLAS::FFLAS\\_UPLO](#) uplo, const [FFLAS::FFLAS\\_TRANSPOSE](#) trans, const [FFLAS::FFLAS\\_DIAG](#) diag, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, size\_t lda, const typename [Field::ConstElement\\_ptr](#) X, size\_t idx)

### 12.30.1 Constructor & Destructor Documentation

#### 12.30.1.1 CheckerImplem\_ftrsm() [1/2]

```
template<class Field >
CheckerImplem_ftrsm (
    const Field & F_,
    const size_t m,
    const size_t n,
    const typename Field::Element alpha,
    const typename Field::ConstElement_ptr B,
    const size_t ldb) [inline]
```

#### 12.30.1.2 CheckerImplem\_ftrsm() [2/2]

```
template<class Field >
CheckerImplem_ftrsm (
    typename Field::RandIter & G,
    const size_t m,
    const size_t n,
    const typename Field::Element alpha,
    const typename Field::ConstElement_ptr B,
    const size_t ldb) [inline]
```

#### 12.30.1.3 ~CheckerImplem\_ftrsm()

```
template<class Field >
~CheckerImplem_ftrsm () [inline]
```



## 12.30.2 Member Function Documentation

### 12.30.2.1 check()

```
template<class Field >
bool check (
    const FFLAS::FFLAS_SIDE side,
    const FFLAS::FFLAS_UPLO uplo,
    const FFLAS::FFLAS_TRANSPOSE trans,
    const FFLAS::FFLAS_DIAG diag,
    const size_t m,
    const size_t n,
    typename Field::Element_ptr A,
    size_t lda,
    const typename Field::ConstElement_ptr X,
    size_t ldx) [inline]
```

The documentation for this class was generated from the following files:

- [checker\\_ftsm.inl](#)
- [checkers\\_fflas.h](#)

## 12.31 CheckerImplem\_invert< Field > Class Template Reference

### Public Member Functions

- [CheckerImplem\\_invert](#) (const [Field](#) &F\_, const size\_t m\_, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda\_)
- [CheckerImplem\\_invert](#) (typename [Field::RandIter](#) &G, const size\_t m\_, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda\_)
- [~CheckerImplem\\_invert](#) ()
- bool [check](#) (typename [Field::ConstElement\\_ptr](#) A, int nullity)

## 12.31.1 Constructor & Destructor Documentation

### 12.31.1.1 CheckerImplem\_invert() [1/2]

```
template<class Field >
CheckerImplem_invert (
    const Field & F_,
    const size_t m_,
    typename Field::ConstElement_ptr A,
    const size_t lda_) [inline]
```

### 12.31.1.2 CheckerImplem\_invert() [2/2]

```
template<class Field >
CheckerImplem_invert (
    typename Field::RandIter & G,
    const size_t m_,
    typename Field::ConstElement_ptr A,
    const size_t lda_) [inline]
```

### 12.31.1.3 ~CheckerImplem\_invert()

```
template<class Field >
~CheckerImplem_invert () [inline]
```

## 12.31.2 Member Function Documentation

### 12.31.2.1 check()

```
template<class Field >
bool check (
    typename Field::ConstElement_ptr A,
    int nullity) [inline]
```

The documentation for this class was generated from the following files:

- [checker\\_invert.inl](#)
- [checkers\\_ffpack.h](#)

## 12.32 CheckerImplem\_PLUQ< Field > Class Template Reference

### Public Member Functions

- [CheckerImplem\\_PLUQ](#) (const [Field](#) &F\_, size\_t m\_, size\_t n\_, typename [Field::ConstElement\\_ptr](#) A, size\_t lda)
- [CheckerImplem\\_PLUQ](#) (typename [Field::RandIter](#) &G, size\_t m\_, size\_t n\_, typename [Field::ConstElement\\_ptr](#) A, size\_t lda)
- [~CheckerImplem\\_PLUQ](#) ()
- bool [check](#) (typename [Field::ConstElement\\_ptr](#) A, size\_t lda, const [FFLAS::FFLAS\\_DIAG](#) Diag, size\_t r, size\_t \*P, size\_t \*Q) const  
*check if the PLUQ factorization is correct.*

## 12.32.1 Constructor & Destructor Documentation

### 12.32.1.1 CheckerImplem\_PLUQ() [1/2]

```
template<class Field >
CheckerImplem_PLUQ (
    const Field & F_,
    size_t m_,
    size_t n_,
    typename Field::ConstElement_ptr A,
    size_t lda) [inline]
```

### 12.32.1.2 CheckerImplem\_PLUQ() [2/2]

```
template<class Field >
CheckerImplem_PLUQ (
    typename Field::RandIter & G,
    size_t m_,
    size_t n_,
    typename Field::ConstElement_ptr A,
    size_t lda) [inline]
```

### 12.32.1.3 ~CheckerImplem\_PLUQ()

```
template<class Field >
~CheckerImplem_PLUQ () [inline]
```

## 12.32.2 Member Function Documentation

### 12.32.2.1 check()

```
template<class Field >
bool check (
    typename Field::ConstElement_ptr A,
```

```

size_t lda,
const FFLAS::FFLAS_DIAG Diag,
size_t r,
size_t * P,
size_t * Q) const [inline]

```

check if the PLUQ factorization is correct.

Returns true if  $w - P(L(U(Q.v))) == 0$

#### Parameters

|          |  |
|----------|--|
| <i>A</i> |  |
| <i>r</i> |  |
| <i>P</i> |  |
| <i>Q</i> |  |

The documentation for this class was generated from the following files:

- [checker\\_pluq.inl](#)
- [checkers\\_ffpack.h](#)

## 12.33 Classic Struct Reference

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

## 12.34 Column Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.35 CompactElement< Element > Struct Template Reference

### Public Types

- typedef Element [type](#)

### 12.35.1 Member Typedef Documentation

#### 12.35.1.1 type

```

template<class Element >
Element type

```

The documentation for this struct was generated from the following file:

- [test-io.C](#)

## 12.36 CompactElement< double > Struct Reference

### Public Types

- typedef int32\_t [type](#)

### 12.36.1 Member Typedef Documentation

#### 12.36.1.1 type

`int32_t` [type](#)

The documentation for this struct was generated from the following file:

- [test-io.C](#)

## 12.37 CompactElement< float > Struct Reference

### Public Types

- `typedef int16_t` [type](#)

### 12.37.1 Member Typedef Documentation

#### 12.37.1.1 type

`int16_t` [type](#)

The documentation for this struct was generated from the following file:

- [test-io.C](#)

## 12.38 CompactElement< int16\_t > Struct Reference

### Public Types

- `typedef int8_t` [type](#)

### 12.38.1 Member Typedef Documentation

#### 12.38.1.1 type

`int8_t` [type](#)

The documentation for this struct was generated from the following file:

- [test-io.C](#)

## 12.39 CompactElement< int32\_t > Struct Reference

### Public Types

- `typedef int16_t` [type](#)

### 12.39.1 Member Typedef Documentation

#### 12.39.1.1 type

`int16_t` [type](#)

The documentation for this struct was generated from the following file:

- [test-io.C](#)

## 12.40 CompactElement< int64\_t > Struct Reference

### Public Types

- `typedef int32_t` [type](#)

## 12.40.1 Member Typedef Documentation

### 12.40.1.1 type

`int32_t type`

The documentation for this struct was generated from the following file:

- [test-io.C](#)

## 12.41 compatible\_data\_type< Field > Struct Template Reference

### Static Public Attributes

- static constexpr bool [value](#) = true

### 12.41.1 Field Documentation

#### 12.41.1.1 value

`template<typename Field >`

`bool value = true [static], [constexpr]`

The documentation for this struct was generated from the following file:

- [benchmark-fgemv.C](#)

## 12.42 compatible\_data\_type< Givaro::ZRing< double > > Struct Reference

### Static Public Attributes

- static constexpr bool [value](#) = false

### 12.42.1 Field Documentation

#### 12.42.1.1 value

`bool value = false [static], [constexpr]`

The documentation for this struct was generated from the following file:

- [benchmark-fgemv.C](#)

## 12.43 compatible\_data\_type< Givaro::ZRing< float > > Struct Reference

### Static Public Attributes

- static constexpr bool [value](#) = false

### 12.43.1 Field Documentation

#### 12.43.1.1 value

`bool value = false [static], [constexpr]`

The documentation for this struct was generated from the following file:

- [benchmark-fgemv.C](#)

## 12.44 `Compose< H1, H2 >` Struct Template Reference

### Public Member Functions

- [Compose](#) ()
- [Compose](#) (const [Compose](#) &*other*)
- [Compose](#) (const [Sequential](#) &*S*)
- [Compose](#) (size\_t *th1*, size\_t *th2*)
- [Compose](#) (const *H1* &*o1*, const *H2* &*o2*)
- *H1* [first\\_component](#) () const
- *H2* [second\\_component](#) () const

### Friends

- std::ostream & [operator<<](#) (std::ostream &*o*, const [Compose](#) &*c*)

### 12.44.1 Constructor & Destructor Documentation

#### 12.44.1.1 `Compose()` [1/5]

```
template<typename H1 = Sequential, typename H2 = Sequential>
Compose () [inline]
```

#### 12.44.1.2 `Compose()` [2/5]

```
template<typename H1 = Sequential, typename H2 = Sequential>
Compose (
    const Compose< H1, H2 > & other) [inline]
```

#### 12.44.1.3 `Compose()` [3/5]

```
template<typename H1 = Sequential, typename H2 = Sequential>
Compose (
    const Sequential & S) [inline]
```

#### 12.44.1.4 `Compose()` [4/5]

```
template<typename H1 = Sequential, typename H2 = Sequential>
Compose (
    size_t th1,
    size_t th2) [inline]
```

#### 12.44.1.5 `Compose()` [5/5]

```
template<typename H1 = Sequential, typename H2 = Sequential>
Compose (
    const H1 & o1,
    const H2 & o2) [inline]
```

### 12.44.2 Member Function Documentation

#### 12.44.2.1 `first_component()`

```
template<typename H1 = Sequential, typename H2 = Sequential>
H1 first\_component () const [inline]
```

#### 12.44.2.2 `second_component()`

```
template<typename H1 = Sequential, typename H2 = Sequential>
H2 second\_component () const [inline]
```

### 12.44.3 Friends And Related Symbol Documentation

#### 12.44.3.1 operator<<

```
template<typename H1 = Sequential, typename H2 = Sequential>
std::ostream & operator<< (
    std::ostream & o,
    const Compose< H1, H2 > & c) [friend]
```

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.45 Simd128\_impl< true, true, false, 2 >::Converter Union Reference

### Data Fields

- [vect\\_t v](#)
- [scalar\\_t t\[vect\\_size\]](#)

### 12.45.1 Field Documentation

#### 12.45.1.1 v

[vect\\_t v](#)

#### 12.45.1.2 t

[scalar\\_t t\[vect\\_size\]](#)

The documentation for this union was generated from the following file:

- [simd128\\_int16.inl](#)

## 12.46 Simd128\_impl< true, true, false, 4 >::Converter Union Reference

### Data Fields

- [vect\\_t v](#)
- [scalar\\_t t\[vect\\_size\]](#)

### 12.46.1 Field Documentation

#### 12.46.1.1 v

[vect\\_t v](#)

#### 12.46.1.2 t

[scalar\\_t t\[vect\\_size\]](#)

The documentation for this union was generated from the following file:

- [simd128\\_int32.inl](#)

## 12.47 Simd128\_impl< true, true, false, 8 >::Converter Union Reference

### Data Fields

- [vect\\_t v](#)
- [scalar\\_t t\[vect\\_size\]](#)

### 12.47.1 Field Documentation

#### 12.47.1.1 `v`

`vect_t v`

#### 12.47.1.2 `t`

`scalar_t t[vect_size]`

The documentation for this union was generated from the following file:

- [simd128\\_int64.inl](#)

## 12.48 `Simd128_impl< true, true, true, 2 >::Converter` Union Reference

### Data Fields

- [vect\\_t v](#)
- [scalar\\_t t\[vect\\_size\]](#)

### 12.48.1 Field Documentation

#### 12.48.1.1 `v`

`vect_t v`

#### 12.48.1.2 `t`

`scalar_t t[vect_size]`

The documentation for this union was generated from the following file:

- [simd128\\_int16.inl](#)

## 12.49 `Simd128_impl< true, true, true, 4 >::Converter` Union Reference

### Data Fields

- [vect\\_t v](#)
- [scalar\\_t t\[vect\\_size\]](#)

### 12.49.1 Field Documentation

#### 12.49.1.1 `v`

`vect_t v`

#### 12.49.1.2 `t`

`scalar_t t[vect_size]`

The documentation for this union was generated from the following file:

- [simd128\\_int32.inl](#)

## 12.50 `Simd128_impl< true, true, true, 8 >::Converter` Union Reference

### Data Fields

- [vect\\_t v](#)
- [scalar\\_t t\[vect\\_size\]](#)



### 12.50.1 Field Documentation

#### 12.50.1.1 v

[vect\\_t](#) v

#### 12.50.1.2 t

[scalar\\_t](#) t[[vect\\_size](#)]

The documentation for this union was generated from the following file:

- [simd128\\_int64.inl](#)

## 12.51 Simd256\_impl< true, false, true, 8 >::Converter Union Reference

### Data Fields

- [vect\\_t](#) v
- [scalar\\_t](#) t[[vect\\_size](#)]

### 12.51.1 Field Documentation

#### 12.51.1.1 v

[vect\\_t](#) v

#### 12.51.1.2 t

[scalar\\_t](#) t[[vect\\_size](#)]

The documentation for this union was generated from the following file:

- [simd256\\_double.inl](#)

## 12.52 Simd256\_impl< true, true, false, 2 >::Converter Union Reference

### Data Fields

- [vect\\_t](#) v
- [scalar\\_t](#) t[[vect\\_size](#)]

### 12.52.1 Field Documentation

#### 12.52.1.1 v

[vect\\_t](#) v

#### 12.52.1.2 t

[scalar\\_t](#) t[[vect\\_size](#)]

The documentation for this union was generated from the following file:

- [simd256\\_int16.inl](#)

## 12.53 Simd256\_impl< true, true, false, 4 >::Converter Union Reference

### Data Fields

- [vect\\_t](#) v
- [scalar\\_t](#) t[[vect\\_size](#)]

### 12.53.1 Field Documentation

#### 12.53.1.1 v

`vect_t v`

#### 12.53.1.2 t

`scalar_t t`

The documentation for this union was generated from the following files:

- [simd256\\_int32.inl](#)
- [simd512\\_int32.inl](#)

## 12.54 Simd256\_impl< true, true, false, 8 >::Converter Union Reference

### Data Fields

- [vect\\_t v](#)
- [scalar\\_t t \[vect\\_size\]](#)

### 12.54.1 Field Documentation

#### 12.54.1.1 v

`vect_t v`

#### 12.54.1.2 t

`scalar_t t [vect_size]`

The documentation for this union was generated from the following file:

- [simd256\\_int64.inl](#)

## 12.55 Simd256\_impl< true, true, true, 2 >::Converter Union Reference

### Data Fields

- [vect\\_t v](#)
- [scalar\\_t t \[vect\\_size\]](#)

### 12.55.1 Field Documentation

#### 12.55.1.1 v

`vect_t v`

#### 12.55.1.2 t

`scalar_t t [vect_size]`

The documentation for this union was generated from the following file:

- [simd256\\_int16.inl](#)

## 12.56 Simd256\_impl< true, true, true, 4 >::Converter Union Reference

### Data Fields

- [vect\\_t v](#)
- [scalar\\_t t \[vect\\_size\]](#)

### 12.56.1 Field Documentation

#### 12.56.1.1 v

`vect_t` v

#### 12.56.1.2 t

`scalar_t` t

The documentation for this union was generated from the following files:

- [simd256\\_int32.inl](#)
- [simd512\\_int32.inl](#)

## 12.57 Simd256\_impl< true, true, true, 8 >::Converter Union Reference

### Data Fields

- `vect_t` v
- `scalar_t` t [vect\_size]

### 12.57.1 Field Documentation

#### 12.57.1.1 v

`vect_t` v

#### 12.57.1.2 t

`scalar_t` t [vect\_size]

The documentation for this union was generated from the following file:

- [simd256\\_int64.inl](#)

## 12.58 Simd512\_impl< true, true, false, 8 >::Converter Union Reference

### Data Fields

- `vect_t` v
- `scalar_t` t [vect\_size]

### 12.58.1 Field Documentation

#### 12.58.1.1 v

`vect_t` v

#### 12.58.1.2 t

`scalar_t` t [vect\_size]

The documentation for this union was generated from the following file:

- [simd512\\_int64.inl](#)

## 12.59 Simd512\_impl< true, true, true, 8 >::Converter Union Reference

### Data Fields

- `vect_t` v
- `scalar_t` t [vect\_size]

## 12.59.1 Field Documentation

### 12.59.1.1 `v`

`vect_t` `v`

### 12.59.1.2 `t`

`scalar_t` `t[vect_size]`

The documentation for this union was generated from the following file:

- [simd512\\_int64.inl](#)

## 12.60 `ConvertTo< T >` Struct Template Reference

Force conversion to appropriate element type of ElementCategory T.

```
#include <field-traits.h>
```

### 12.60.1 Detailed Description

```
template<class T>
```

```
struct FFLAS::ModeCategories::ConvertTo< T >
```

Force conversion to appropriate element type of ElementCategory T.

e.g.

- `ConvertTo<ElementCategories::MachineFloatTag>` tries conversion of `Modular<int>` to `Modular<double>`
- `ConvertTo<ElementCategories::FixedPreIntTag>` tries conversion of `Modular<Integer>` to `Modular<RecInt<K>>`
- `ConvertTo<ElementCategories::ArbitraryPreIntTag>` tries conversion of `Modular<Integer>` to `RNSInteger`

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.61 `Coo< ValT, IdxT >` Struct Template Reference

### Public Types

- using `Self` = `Coo<ValT, IdxT>`

### Public Member Functions

- `Coo` (`ValT` `v`, `IdxT` `r`, `IdxT` `c`)
- `Coo` ()=default
- `Coo` (const `Self` &)=default
- `Coo` (`Self` &&)=default
- `Self` & `operator=` (const `Self` &)=default
- `Self` & `operator=` (`Self` &&)=default

### Data Fields

- `ValT` `val` = 0
- `IdxT` `row` = 0
- `IdxT` `col` = 0

## 12.61.1 Member Typedef Documentation

### 12.61.1.1 Self

```
template<class ValT , class IdxT >
using Self = Coo<ValT, IdxT>
```

## 12.61.2 Constructor & Destructor Documentation

### 12.61.2.1 Coo() [1/4]

```
template<class ValT , class IdxT >
Coo (
    ValT v,
    IdxT r,
    IdxT c) [inline]
```

### 12.61.2.2 Coo() [2/4]

```
template<class ValT , class IdxT >
Coo () [default]
```

### 12.61.2.3 Coo() [3/4]

```
template<class ValT , class IdxT >
Coo (
    const Self & ) [default]
```

### 12.61.2.4 Coo() [4/4]

```
template<class ValT , class IdxT >
Coo (
    Self && ) [default]
```

## 12.61.3 Member Function Documentation

### 12.61.3.1 operator=() [1/2]

```
template<class ValT , class IdxT >
Self & operator= (
    const Self & ) [default]
```

### 12.61.3.2 operator=() [2/2]

```
template<class ValT , class IdxT >
Self & operator= (
    Self && ) [default]
```

## 12.61.4 Field Documentation

### 12.61.4.1 val

```
template<class ValT , class IdxT >
ValT val = 0
```

### 12.61.4.2 row

```
template<class ValT , class IdxT >
IdxT row = 0
```

### 12.61.4.3 col

```
template<class ValT , class IdxT >
IdxT col = 0
```

The documentation for this struct was generated from the following file:

- [csr\\_hyb\\_utils.inl](#)

## 12.62 **Coo**< **Field** > Struct Template Reference

```
#include <read_sparse.h>
```

### Public Member Functions

- **Coo** ()=default
- **Coo** (typename **Field**::**Element** v, **index\_t** r, **index\_t** c)
- **Coo** (const **Self** &)=default
- **Coo** (**Self** &&)=default
- **Self** & **operator=** (const **Self** &)=default
- **Self** & **operator=** (**Self** &&)=default

### Data Fields

- **Field**::**Element** val = 0
- **index\_t** col = 0
- **index\_t** row = 0
- bool **deleted** = false

## 12.62.1 Constructor & Destructor Documentation

### 12.62.1.1 **Coo**() [1/4]

```
template<class Field >
Coo () [default]
```

### 12.62.1.2 **Coo**() [2/4]

```
template<class Field >
Coo (
    typename Field::Element v,
    index_t r,
    index_t c) [inline]
```

### 12.62.1.3 **Coo**() [3/4]

```
template<class Field >
Coo (
    const Self & ) [default]
```

### 12.62.1.4 **Coo**() [4/4]

```
template<class Field >
Coo (
    Self && ) [default]
```

## 12.62.2 Member Function Documentation

### 12.62.2.1 `operator=()` [1/2]

```
template<class Field >
Self & operator= (
    const Self & ) [default]
```

### 12.62.2.2 `operator=()` [2/2]

```
template<class Field >
Self & operator= (
    Self && ) [default]
```

## 12.62.3 Field Documentation

### 12.62.3.1 `val`

```
template<class Field >
Field::Element val = 0
```

### 12.62.3.2 `col`

```
template<class Field >
index_t col = 0
```

### 12.62.3.3 `row`

```
template<class Field >
index_t row = 0
```

### 12.62.3.4 `deleted`

```
template<class Field >
bool deleted = false
```

The documentation for this struct was generated from the following file:

- [read\\_sparse.h](#)

## 12.63 `Coo< ValT, IdxT >` Struct Template Reference

### Public Types

- using `Self` = `Coo<ValT, IdxT>`

### Public Member Functions

- `Coo` (`ValT` v, `IdxT` r, `IdxT` c)
- `Coo` ()=default
- `Coo` (const `Self` &)=default
- `Coo` (`Self` &&)=default
- `Self` & `operator=` (const `Self` &)=default
- `Self` & `operator=` (`Self` &&)=default

### Data Fields

- `ValT` `val` = 0
- `IdxT` `row` = 0
- `IdxT` `col` = 0

## 12.63.1 Member Typedef Documentation

### 12.63.1.1 Self

```
template<class ValT , class IdxT >
using Self = Co<ValT, IdxT>
```

## 12.63.2 Constructor & Destructor Documentation

### 12.63.2.1 Co() [1/4]

```
template<class ValT , class IdxT >
Co (
    ValT v,
    IdxT r,
    IdxT c) [inline]
```

### 12.63.2.2 Co() [2/4]

```
template<class ValT , class IdxT >
Co () [default]
```

### 12.63.2.3 Co() [3/4]

```
template<class ValT , class IdxT >
Co (
    const Self & ) [default]
```

### 12.63.2.4 Co() [4/4]

```
template<class ValT , class IdxT >
Co (
    Self && ) [default]
```

## 12.63.3 Member Function Documentation

### 12.63.3.1 operator=() [1/2]

```
template<class ValT , class IdxT >
Self & operator= (
    const Self & ) [default]
```

### 12.63.3.2 operator=() [2/2]

```
template<class ValT , class IdxT >
Self & operator= (
    Self && ) [default]
```

## 12.63.4 Field Documentation

### 12.63.4.1 val

```
template<class ValT , class IdxT >
ValT val = 0
```

### 12.63.4.2 row

```
template<class ValT , class IdxT >
IdxT row = 0
```



**12.63.4.3 col**

```
template<class ValT , class IdxT >
IdxT col = 0
```

The documentation for this struct was generated from the following file:

- [sell\\_utils.inl](#)

**12.64 CooMat< Field > Struct Template Reference**

```
#include <fflas_sparse.h>
```

**Data Fields**

- [FFLAS::Sparse< Field, SparseMatrix\\_t::COO, int16\\_t > \\* \\_coo16](#) = nullptr
- [FFLAS::Sparse< Field, SparseMatrix\\_t::COO, int32\\_t > \\* \\_coo32](#) = nullptr
- [FFLAS::Sparse< Field, SparseMatrix\\_t::COO, int64\\_t > \\* \\_coo64](#) = nullptr
- [FFLAS::Sparse< Field, SparseMatrix\\_t::COO\\_ZO, int16\\_t > \\* \\_coo16\\_zo](#) = nullptr
- [FFLAS::Sparse< Field, SparseMatrix\\_t::COO\\_ZO, int32\\_t > \\* \\_coo32\\_zo](#) = nullptr
- [FFLAS::Sparse< Field, SparseMatrix\\_t::COO\\_ZO, int64\\_t > \\* \\_coo64\\_zo](#) = nullptr

**12.64.1 Field Documentation****12.64.1.1 \_coo16**

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::COO, int16_t>* _coo16 = nullptr
```

**12.64.1.2 \_coo32**

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::COO, int32_t>* _coo32 = nullptr
```

**12.64.1.3 \_coo64**

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::COO, int64_t>* _coo64 = nullptr
```

**12.64.1.4 \_coo16\_zo**

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::COO_ZO, int16_t>* _coo16_zo = nullptr
```

**12.64.1.5 \_coo32\_zo**

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::COO_ZO, int32_t>* _coo32_zo = nullptr
```

**12.64.1.6 \_coo64\_zo**

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::COO_ZO, int64_t>* _coo64_zo = nullptr
```

The documentation for this struct was generated from the following file:

- [fflas\\_sparse.h](#)

## 12.65 `count_nonconst_lvalue_reference< T >` Struct Template Reference

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.66 `count_nonconst_lvalue_reference< const T &, O... >` Struct Template Reference

### Static Public Attributes

- static constexpr `size_t` `n` = `count_nonconst_lvalue_reference<O...>::n`

### 12.66.1 Field Documentation

#### 12.66.1.1 `n`

```
template<typename T , typename... O>
size_t n = count_nonconst_lvalue_reference<O...>::n [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.67 `count_nonconst_lvalue_reference< T &, O... >` Struct Template Reference

### Static Public Attributes

- static constexpr `size_t` `n`

### 12.67.1 Field Documentation

#### 12.67.1.1 `n`

```
template<typename T , typename... O>
size_t n [static], [constexpr]
Initial value:
= std::integral_constant<size_t, 1>::value
+ count_nonconst_lvalue_reference<O...>::n
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.68 `count_nonconst_lvalue_reference< T, O... >` Struct Template Reference

### Static Public Attributes

- static constexpr `size_t` `n` = `count_nonconst_lvalue_reference<O...>::n`

### 12.68.1 Field Documentation

#### 12.68.1.1 `n`

```
template<typename T , typename... O>
size_t n = count_nonconst_lvalue_reference<O...>::n [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.69 count\_nonconst\_lvalue\_reference<> Struct Reference

### Static Public Attributes

- static constexpr size\_t [n](#) = std::integral\_constant<size\_t, 0>::value

### 12.69.1 Field Documentation

#### 12.69.1.1 [n](#)

size\_t [n](#) = std::integral\_constant<size\_t, 0>::value [static], [constexpr]

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.70 CsrMat< Field > Struct Template Reference

```
#include <fflas_sparse.h>
```

### Data Fields

- [FFLAS::Sparse< Field, SparseMatrix\\_t::CSR, int16\\_t > \\* \\_csr16](#) = nullptr
- [FFLAS::Sparse< Field, SparseMatrix\\_t::CSR, int32\\_t > \\* \\_csr32](#) = nullptr
- [FFLAS::Sparse< Field, SparseMatrix\\_t::CSR, int64\\_t > \\* \\_csr64](#) = nullptr
- [FFLAS::Sparse< Field, SparseMatrix\\_t::CSR\\_ZO, int16\\_t > \\* \\_csr16\\_zo](#) = nullptr
- [FFLAS::Sparse< Field, SparseMatrix\\_t::CSR\\_ZO, int32\\_t > \\* \\_csr32\\_zo](#) = nullptr
- [FFLAS::Sparse< Field, SparseMatrix\\_t::CSR\\_ZO, int64\\_t > \\* \\_csr64\\_zo](#) = nullptr

### 12.70.1 Field Documentation

#### 12.70.1.1 [\\_csr16](#)

```
template<class Field >
```

```
FFLAS::Sparse<Field, SparseMatrix_t::CSR, int16_t>* _csr16 = nullptr
```

#### 12.70.1.2 [\\_csr32](#)

```
template<class Field >
```

```
FFLAS::Sparse<Field, SparseMatrix_t::CSR, int32_t>* _csr32 = nullptr
```

#### 12.70.1.3 [\\_csr64](#)

```
template<class Field >
```

```
FFLAS::Sparse<Field, SparseMatrix_t::CSR, int64_t>* _csr64 = nullptr
```

#### 12.70.1.4 [\\_csr16\\_zo](#)

```
template<class Field >
```

```
FFLAS::Sparse<Field, SparseMatrix_t::CSR_ZO, int16_t>* _csr16_zo = nullptr
```

#### 12.70.1.5 [\\_csr32\\_zo](#)

```
template<class Field >
```

```
FFLAS::Sparse<Field, SparseMatrix_t::CSR_ZO, int32_t>* _csr32_zo = nullptr
```

#### 12.70.1.6 [\\_csr64\\_zo](#)

```
template<class Field >
```

```
FFLAS::Sparse<Field, SparseMatrix_t::CSR_ZO, int64_t>* _csr64_zo = nullptr
```

The documentation for this struct was generated from the following file:

- [fflas\\_sparse.h](#)

## 12.71 DefaultBoundedTag Struct Reference

Use standard field operations, but keeps track of bounds on input and output.

```
#include <field-traits.h>
```

### 12.71.1 Detailed Description

Use standard field operations, but keeps track of bounds on input and output.

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.72 DefaultTag Struct Reference

No specific mode of action: use standard field operations.

```
#include <field-traits.h>
```

### 12.72.1 Detailed Description

No specific mode of action: use standard field operations.

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.73 DelayedTag Struct Reference

Performs field operations with delayed mod reductions. Ensures result is reduced.

```
#include <field-traits.h>
```

### 12.73.1 Detailed Description

Performs field operations with delayed mod reductions. Ensures result is reduced.

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.74 DivideAndConquer Struct Reference

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

## 12.75 ElementTraits< Element > Struct Template Reference

[ElementTraits](#).

```
#include <field-traits.h>
```

### Public Types

- typedef [ElementCategories::GenericTag](#) value

### 12.75.1 Detailed Description

```
template<class Element>
```

```
struct FFLAS::ElementTraits< Element >
```

[ElementTraits](#).

## 12.75.2 Member Typedef Documentation

### 12.75.2.1 value

```
template<class Element >
```

```
ElementCategories::GenericTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.76 ElementTraits< double > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ElementCategories::MachineFloatTag](#) value

## 12.76.1 Member Typedef Documentation

### 12.76.1.1 value

```
ElementCategories::MachineFloatTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.77 ElementTraits< FFPACK::rns\_double\_elt > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ElementCategories::RNSElementTag](#) value

## 12.77.1 Member Typedef Documentation

### 12.77.1.1 value

```
ElementCategories::RNSElementTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.78 ElementTraits< float > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ElementCategories::MachineFloatTag](#) value

## 12.78.1 Member Typedef Documentation

### 12.78.1.1 value

```
ElementCategories::MachineFloatTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.79 ElementTraits< Givaro::Integer > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ElementCategories::ArbitraryPrecIntTag](#) value

### 12.79.1 Member Typedef Documentation

#### 12.79.1.1 value

[ElementCategories::ArbitraryPrecIntTag](#) value

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.80 ElementTraits< int16\_t > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ElementCategories::MachineIntTag](#) value

### 12.80.1 Member Typedef Documentation

#### 12.80.1.1 value

[ElementCategories::MachineIntTag](#) value

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.81 ElementTraits< int32\_t > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ElementCategories::MachineIntTag](#) value

### 12.81.1 Member Typedef Documentation

#### 12.81.1.1 value

[ElementCategories::MachineIntTag](#) value

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.82 ElementTraits< int64\_t > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ElementCategories::MachineIntTag](#) value

## 12.82.1 Member Typedef Documentation

### 12.82.1.1 value

[ElementCategories::MachineIntTag](#) value

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.83 ElementTraits< int8\_t > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ElementCategories::MachineIntTag](#) value

## 12.83.1 Member Typedef Documentation

### 12.83.1.1 value

[ElementCategories::MachineIntTag](#) value

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.84 ElementTraits< RecInt::rint< K > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ElementCategories::FixedPrecIntTag](#) value

## 12.84.1 Member Typedef Documentation

### 12.84.1.1 value

```
template<size_t K>
```

[ElementCategories::FixedPrecIntTag](#) value

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.85 ElementTraits< RecInt::rmint< K, MG > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ElementCategories::FixedPrecIntTag](#) value

## 12.85.1 Member Typedef Documentation

### 12.85.1.1 value

```
template<size_t K, int MG>
```

[ElementCategories::FixedPrecIntTag](#) value

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.86 ElementTraits< Reclnt::ruint< K > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ElementCategories::FixedPrecIntTag](#) value

### 12.86.1 Member Typedef Documentation

#### 12.86.1.1 value

```
template<size_t K>
```

```
ElementCategories::FixedPrecIntTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.87 ElementTraits< uint16\_t > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ElementCategories::MachineIntTag](#) value

### 12.87.1 Member Typedef Documentation

#### 12.87.1.1 value

```
ElementCategories::MachineIntTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.88 ElementTraits< uint32\_t > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ElementCategories::MachineIntTag](#) value

### 12.88.1 Member Typedef Documentation

#### 12.88.1.1 value

```
ElementCategories::MachineIntTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.89 ElementTraits< uint64\_t > Struct Reference

```
#include <field-traits.h>
```



**Public Types**

- typedef [ElementCategories::MachineIntTag](#) value

**12.89.1 Member Typedef Documentation****12.89.1.1 value**

[ElementCategories::MachineIntTag](#) value

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

**12.90 ElementTraits< uint8\_t > Struct Reference**

```
#include <field-traits.h>
```

**Public Types**

- typedef [ElementCategories::MachineIntTag](#) value

**12.90.1 Member Typedef Documentation****12.90.1.1 value**

[ElementCategories::MachineIntTag](#) value

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

**12.91 EILMat< Field > Struct Template Reference**

```
#include <fflas_sparse.h>
```

**Data Fields**

- [FFLAS::Sparse< Field, SparseMatrix\\_t::ELL, int16\\_t > \\* \\_ell16](#) = nullptr
- [FFLAS::Sparse< Field, SparseMatrix\\_t::ELL, int32\\_t > \\* \\_ell32](#) = nullptr
- [FFLAS::Sparse< Field, SparseMatrix\\_t::ELL, int64\\_t > \\* \\_ell64](#) = nullptr
- [FFLAS::Sparse< Field, SparseMatrix\\_t::ELL\\_ZO, int16\\_t > \\* \\_ell16\\_zo](#) = nullptr
- [FFLAS::Sparse< Field, SparseMatrix\\_t::ELL\\_ZO, int32\\_t > \\* \\_ell32\\_zo](#) = nullptr
- [FFLAS::Sparse< Field, SparseMatrix\\_t::ELL\\_ZO, int64\\_t > \\* \\_ell64\\_zo](#) = nullptr

**12.91.1 Field Documentation****12.91.1.1 \_ell16**

```
template<class Field >
```

```
FFLAS::Sparse<Field, SparseMatrix_t::ELL, int16_t>* _ell16 = nullptr
```

**12.91.1.2 \_ell32**

```
template<class Field >
```

```
FFLAS::Sparse<Field, SparseMatrix_t::ELL, int32_t>* _ell32 = nullptr
```

**12.91.1.3 \_ell64**

```
template<class Field >
```

```
FFLAS::Sparse<Field, SparseMatrix_t::ELL, int64_t>* _ell64 = nullptr
```

#### 12.91.1.4 `_ell16_zo`

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::ELL_ZO, int16_t>* _ell16_zo = nullptr
```

#### 12.91.1.5 `_ell32_zo`

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::ELL_ZO, int32_t>* _ell32_zo = nullptr
```

#### 12.91.1.6 `_ell64_zo`

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::ELL_ZO, int64_t>* _ell64_zo = nullptr
```

The documentation for this struct was generated from the following file:

- [fflas\\_sparse.h](#)

## 12.92 Failure Class Reference

A precondition failed.

```
#include <debug.h>
```

### Public Member Functions

- [Failure](#) ()
- void [operator\(\)](#) (const char \*function, int line, const char \*check)
- void [operator\(\)](#) (const char \*function, const char \*file, int line, const char \*check)
- void [setErrorStream](#) (std::ostream &stream)
- std::ostream & [print](#) (std::ostream &o) const

### Protected Attributes

- std::ostream \* [\\_errorStream](#)

### 12.92.1 Detailed Description

A precondition failed.

The `throw` mechanism is usually used here as in

```
if (!check)
failure()(__func__, __LINE__, "this check just failed");
```

The parameters of the constructor help debugging.

### 12.92.2 Constructor & Destructor Documentation

#### 12.92.2.1 `Failure()`

```
Failure () [inline]
```

### 12.92.3 Member Function Documentation

#### 12.92.3.1 `operator>()` [1/2]

```
void operator() (
    const char * function,
    int line,
    const char * check) [inline]
```

A precondition failed.

## Parameters

|                 |                                                                   |
|-----------------|-------------------------------------------------------------------|
| <i>function</i> | usually <code>__func__</code> , the function that threw the error |
| <i>line</i>     | usually <code>__LINE__</code> , the line where it happened        |
| <i>check</i>    | a string telling what failed.                                     |

**12.92.3.2 operator>() [2/2]**

```
void operator() (
    const char * function,
    const char * file,
    int line,
    const char * check) [inline]
```

A precondition failed. The parameter help debugging. This is not much different from the previous except we can dig faster in the file where the exception was triggered.

## Parameters

|                 |                                                                   |
|-----------------|-------------------------------------------------------------------|
| <i>function</i> | usually <code>__func__</code> , the function that threw the error |
| <i>file</i>     | usually <code>__FILE__</code> , the file where this function is   |
| <i>line</i>     | usually <code>__LINE__</code> , the line where it happened        |
| <i>check</i>    | a string telling what failed.                                     |

**12.92.3.3 setErrorMessage()**

```
void setErrorMessage (
    std::ostream & stream)
```

**12.92.3.4 print()**

```
std::ostream & print (
    std::ostream & o) const [inline]
```

overload the virtual print of LinboxError.

## Parameters

|          |               |
|----------|---------------|
| <i>o</i> | output stream |
|----------|---------------|

**12.92.4 Field Documentation****12.92.4.1 \_errorMessage**

```
std::ostream* _errorMessage [protected]
```

The documentation for this class was generated from the following file:

- [debug.h](#)

**12.93 FailureCharpolyCheck Class Reference**

```
#include <checkers_ffpack.h>
```

The documentation for this class was generated from the following file:

- [checkers\\_ffpack.h](#)

## 12.94 FailureDetCheck Class Reference

```
#include <checkers_ffpack.h>
```

The documentation for this class was generated from the following file:

- [checkers\\_ffpack.h](#)

## 12.95 FailureFgemmCheck Class Reference

```
#include <checkers_fflas.h>
```

The documentation for this class was generated from the following file:

- [checkers\\_fflas.h](#)

## 12.96 FailureInvertCheck Class Reference

```
#include <checkers_ffpack.h>
```

The documentation for this class was generated from the following file:

- [checkers\\_ffpack.h](#)

## 12.97 FailurePLUQCheck Class Reference

```
#include <checkers_ffpack.h>
```

The documentation for this class was generated from the following file:

- [checkers\\_ffpack.h](#)

## 12.98 FailureTrsmCheck Class Reference

```
#include <checkers_fflas.h>
```

The documentation for this class was generated from the following file:

- [checkers\\_fflas.h](#)

## 12.99 FieldSimd<\_Field> Class Template Reference

### Public Types

- using [Field](#) = [\\_Field](#)
- using [Element](#) = typename [Field::Element](#)
- using [simd](#) = [Simd](#)<typename [\\_Field::Element](#)>
- using [vect\\_t](#) = typename [simd::vect\\_t](#)
- using [scalar\\_t](#) = typename [simd::scalar\\_t](#)

### Public Member Functions

- [FieldSimd](#) (const [Field](#) &f)
- [FieldSimd](#) (const [Self](#) &)=default
- [FieldSimd](#) ([Self](#) &&)=default
- [Self](#) & operator= (const [Self](#) &)=default
- [Self](#) & operator= ([Self](#) &&)=default
- [INLINE vect\\_t](#) init ([vect\\_t](#) &x, const [vect\\_t](#) a) const
- [INLINE vect\\_t](#) init (const [vect\\_t](#) a) const
- [INLINE vect\\_t](#) add ([vect\\_t](#) &c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- [INLINE vect\\_t](#) add (const [vect\\_t](#) a, const [vect\\_t](#) b)

- `INLINE vect_t addin (vect_t &a, const vect_t b) const`
- `INLINE vect_t add_r (vect_t &c, const vect_t a, const vect_t b) const`
- `INLINE vect_t add_r (const vect_t a, const vect_t b) const`
- `INLINE vect_t addin_r (vect_t &a, const vect_t b) const`
- `INLINE vect_t sub (vect_t &c, const vect_t a, const vect_t b)`
- `INLINE vect_t sub (const vect_t a, const vect_t b)`
- `INLINE vect_t subin (vect_t &a, const vect_t b) const`
- `INLINE vect_t sub_r (vect_t &c, const vect_t a, const vect_t b) const`
- `INLINE vect_t sub_r (const vect_t a, const vect_t b) const`
- `INLINE vect_t subin_r (vect_t &a, const vect_t b) const`
- `INLINE vect_t zero (vect_t &x) const`
- `INLINE vect_t zero () const`
- `INLINE vect_t mod (vect_t &c) const`
- `INLINE vect_t mul (vect_t &c, const vect_t a, const vect_t b) const`
- `INLINE vect_t mul (const vect_t a, const vect_t b) const`
- `INLINE vect_t mulin (vect_t &a, const vect_t b) const`
- `INLINE vect_t mul_r (vect_t &c, const vect_t a, const vect_t b) const`
- `INLINE vect_t mul_r (const vect_t a, const vect_t b) const`
- `INLINE vect_t axpy (vect_t &r, const vect_t a, const vect_t b, const vect_t c) const`
- `INLINE vect_t axpy (const vect_t c, const vect_t a, const vect_t b) const`
- `INLINE vect_t axpyin (vect_t &c, const vect_t a, const vect_t b) const`
- `INLINE vect_t axpy_r (vect_t &r, const vect_t a, const vect_t b, const vect_t c) const`
- `INLINE vect_t axpy_r (const vect_t c, const vect_t a, const vect_t b) const`
- `INLINE vect_t axpyin_r (vect_t &c, const vect_t a, const vect_t b) const`
- `INLINE vect_t maxpy (vect_t &r, const vect_t a, const vect_t b, const vect_t c) const`
- `INLINE vect_t maxpy (const vect_t c, const vect_t a, const vect_t b) const`
- `INLINE vect_t maxpyin (vect_t &c, const vect_t a, const vect_t b) const`

### Static Public Attributes

- static const constexpr size\_t `vect_size` = `simd::vect_size`
- static const constexpr size\_t `alignment` = `simd::alignment`

## 12.99.1 Member Typedef Documentation

### 12.99.1.1 Field

```
template<class _Field >
using Field = _Field
```

### 12.99.1.2 Element

```
template<class _Field >
using Element = typename Field::Element
```

### 12.99.1.3 simd

```
template<class _Field >
using simd = Simd<typename _Field::Element>
```

### 12.99.1.4 vect\_t

```
template<class _Field >
using vect_t = typename simd::vect_t
```

### 12.99.1.5 scalar\_t

```
template<class _Field >
using scalar_t = typename simd::scalar_t
```

## 12.99.2 Constructor & Destructor Documentation

### 12.99.2.1 FieldSimd() [1/3]

```
template<class _Field >
FieldSimd (
    const Field & f) [inline]
```

### 12.99.2.2 FieldSimd() [2/3]

```
template<class _Field >
FieldSimd (
    const Self & ) [default]
```

### 12.99.2.3 FieldSimd() [3/3]

```
template<class _Field >
FieldSimd (
    Self && ) [default]
```

## 12.99.3 Member Function Documentation

### 12.99.3.1 operator=() [1/2]

```
template<class _Field >
Self & operator= (
    const Self & ) [default]
```

### 12.99.3.2 operator=() [2/2]

```
template<class _Field >
Self & operator= (
    Self && ) [default]
```

### 12.99.3.3 init() [1/2]

```
template<class _Field >
INLINE vect_t init (
    vect_t & x,
    const vect_t a) const [inline]
```

### 12.99.3.4 init() [2/2]

```
template<class _Field >
INLINE vect_t init (
    const vect_t a) const [inline]
```

### 12.99.3.5 add() [1/2]

```
template<class _Field >
INLINE vect_t add (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline]
```

**12.99.3.6 add()** [2/2]

```
template<class _Field >
INLINE vect_t add (
    const vect_t a,
    const vect_t b) [inline]
```

**12.99.3.7 addin()**

```
template<class _Field >
INLINE vect_t addin (
    vect_t & a,
    const vect_t b) const [inline]
```

**12.99.3.8 add\_r()** [1/2]

```
template<class _Field >
INLINE vect_t add_r (
    vect_t & c,
    const vect_t a,
    const vect_t b) const [inline]
```

**12.99.3.9 add\_r()** [2/2]

```
template<class _Field >
INLINE vect_t add_r (
    const vect_t a,
    const vect_t b) const [inline]
```

**12.99.3.10 addin\_r()**

```
template<class _Field >
INLINE vect_t addin_r (
    vect_t & a,
    const vect_t b) const [inline]
```

**12.99.3.11 sub()** [1/2]

```
template<class _Field >
INLINE vect_t sub (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline]
```

**12.99.3.12 sub()** [2/2]

```
template<class _Field >
INLINE vect_t sub (
    const vect_t a,
    const vect_t b) [inline]
```

**12.99.3.13 subin()**

```
template<class _Field >
INLINE vect_t subin (
    vect_t & a,
    const vect_t b) const [inline]
```

**12.99.3.14 sub\_r()** [1/2]

```
template<class _Field >
INLINE vect_t sub_r (
    vect_t & c,
    const vect_t a,
    const vect_t b) const [inline]
```

**12.99.3.15 sub\_r()** [2/2]

```
template<class _Field >
INLINE vect_t sub_r (
    const vect_t a,
    const vect_t b) const [inline]
```

**12.99.3.16 subin\_r()**

```
template<class _Field >
INLINE vect_t subin_r (
    vect_t & a,
    const vect_t b) const [inline]
```

**12.99.3.17 zero()** [1/2]

```
template<class _Field >
INLINE vect_t zero (
    vect_t & x) const [inline]
```

**12.99.3.18 zero()** [2/2]

```
template<class _Field >
INLINE vect_t zero () const [inline]
```

**12.99.3.19 mod()**

```
template<class _Field >
INLINE vect_t mod (
    vect_t & c) const [inline]
```

**12.99.3.20 mul()** [1/2]

```
template<class _Field >
INLINE vect_t mul (
    vect_t & c,
    const vect_t a,
    const vect_t b) const [inline]
```

**12.99.3.21 mul()** [2/2]

```
template<class _Field >
INLINE vect_t mul (
    const vect_t a,
    const vect_t b) const [inline]
```

**12.99.3.22 mulin()**

```
template<class _Field >
INLINE vect_t mulin (
    vect_t & a,
    const vect_t b) const [inline]
```



**12.99.3.23 mul\_r() [1/2]**

```
template<class _Field >
INLINE vect_t mul_r (
    vect_t & c,
    const vect_t a,
    const vect_t b) const [inline]
```

**12.99.3.24 mul\_r() [2/2]**

```
template<class _Field >
INLINE vect_t mul_r (
    const vect_t a,
    const vect_t b) const [inline]
```

**12.99.3.25 axpy() [1/2]**

```
template<class _Field >
INLINE vect_t axpy (
    vect_t & r,
    const vect_t a,
    const vect_t b,
    const vect_t c) const [inline]
```

**12.99.3.26 axpy() [2/2]**

```
template<class _Field >
INLINE vect_t axpy (
    const vect_t c,
    const vect_t a,
    const vect_t b) const [inline]
```

**12.99.3.27 axpyin()**

```
template<class _Field >
INLINE vect_t axpyin (
    vect_t & c,
    const vect_t a,
    const vect_t b) const [inline]
```

**12.99.3.28 axpy\_r() [1/2]**

```
template<class _Field >
INLINE vect_t axpy_r (
    vect_t & r,
    const vect_t a,
    const vect_t b,
    const vect_t c) const [inline]
```

**12.99.3.29 axpy\_r() [2/2]**

```
template<class _Field >
INLINE vect_t axpy_r (
    const vect_t c,
    const vect_t a,
    const vect_t b) const [inline]
```

**12.99.3.30 axpyin\_r()**

```
template<class _Field >
INLINE vect_t axpyin_r (
    vect_t & c,
    const vect_t a,
    const vect_t b) const [inline]
```

**12.99.3.31 maxpy() [1/2]**

```
template<class _Field >
INLINE vect_t maxpy (
    vect_t & r,
    const vect_t a,
    const vect_t b,
    const vect_t c) const [inline]
```

**12.99.3.32 maxpy() [2/2]**

```
template<class _Field >
INLINE vect_t maxpy (
    const vect_t c,
    const vect_t a,
    const vect_t b) const [inline]
```

**12.99.3.33 maxpyin()**

```
template<class _Field >
INLINE vect_t maxpyin (
    vect_t & c,
    const vect_t a,
    const vect_t b) const [inline]
```

**12.99.4 Field Documentation****12.99.4.1 vect\_size**

```
template<class _Field >
const constexpr size_t vect_size = simd::vect_size [static], [constexpr]
```

**12.99.4.2 alignment**

```
template<class _Field >
const constexpr size_t alignment = simd::alignment [static], [constexpr]
```

The documentation for this class was generated from the following file:

- [simd\\_modular.inl](#)

**12.100 FieldTraits< Field > Struct Template Reference**

FieldTrait.

```
#include <field-traits.h>
```

**Public Types**

- typedef [FieldCategories::GenericTag](#) category

**Static Public Attributes**

- static const bool [balanced](#) = false

### 12.100.1 Detailed Description

```
template<class Field>
struct FFLAS::FieldTraits< Field >
```

FieldTrait.

### 12.100.2 Member Typedef Documentation

#### 12.100.2.1 category

```
template<class Field >
FieldCategories::GenericTag category
```

### 12.100.3 Field Documentation

#### 12.100.3.1 balanced

```
template<class Field >
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.101 FieldTraits< FFPACK::RNSInteger< T > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [FieldCategories::UnparametricTag](#) category

### Static Public Attributes

- static const bool [balanced](#) = false

### 12.101.1 Member Typedef Documentation

#### 12.101.1.1 category

```
template<typename T >
FieldCategories::UnparametricTag category
```

### 12.101.2 Field Documentation

#### 12.101.2.1 balanced

```
template<typename T >
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.102 FieldTraits< FFPACK::RNSIntegerMod< T > > Struct Template Reference

```
#include <field-traits.h>
```

## Public Types

- typedef [FieldCategories::ModularTag](#) category

## Static Public Attributes

- static const bool [balanced](#) = false

### 12.102.1 Member Typedef Documentation

#### 12.102.1.1 category

```
template<typename T >
FieldCategories::ModularTag category
```

### 12.102.2 Field Documentation

#### 12.102.2.1 balanced

```
template<typename T >
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.103 FieldTraits< Givaro::Modular< Element > > Struct Template Reference

```
#include <field-traits.h>
```

## Public Types

- typedef [FieldCategories::ModularTag](#) category

## Static Public Attributes

- static const bool [balanced](#) = false

### 12.103.1 Member Typedef Documentation

#### 12.103.1.1 category

```
template<class Element >
FieldCategories::ModularTag category
```

### 12.103.2 Field Documentation

#### 12.103.2.1 balanced

```
template<class Element >
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.104 FieldTraits< Givaro::ModularBalanced< Element > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [FieldCategories::ModularTag](#) category

### Static Public Attributes

- static const bool [balanced](#) = true

## 12.104.1 Member Typedef Documentation

### 12.104.1.1 category

```
template<class Element >  
FieldCategories::ModularTag category
```

## 12.104.2 Field Documentation

### 12.104.2.1 balanced

```
template<class Element >  
const bool balanced = true [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.105 FieldTraits< Givaro::ZRing< double > > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [FieldCategories::UnparametricTag](#) category

### Static Public Attributes

- static const bool [balanced](#) = false

## 12.105.1 Member Typedef Documentation

### 12.105.1.1 category

```
FieldCategories::UnparametricTag category
```

## 12.105.2 Field Documentation

### 12.105.2.1 balanced

```
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.106 FieldTraits< Givaro::ZRing< float > > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [FieldCategories::UnparametricTag](#) category

**Static Public Attributes**

- static const bool [balanced](#) = false

**12.106.1 Member Typedef Documentation****12.106.1.1 category**

[FieldCategories::UnparametricTag category](#)

**12.106.2 Field Documentation****12.106.2.1 balanced**

```
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

**12.107 FieldTraits< Givaro::ZRing< Givaro::Integer > > Struct Reference**

```
#include <field-traits.h>
```

**Public Types**

- typedef [FieldCategories::UnparametricTag category](#)

**Static Public Attributes**

- static const bool [balanced](#) = false

**12.107.1 Member Typedef Documentation****12.107.1.1 category**

[FieldCategories::UnparametricTag category](#)

**12.107.2 Field Documentation****12.107.2.1 balanced**

```
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

**12.108 FieldTraits< Givaro::ZRing< int16\_t > > Struct Reference**

```
#include <field-traits.h>
```

**Public Types**

- typedef [FieldCategories::UnparametricTag category](#)

**Static Public Attributes**

- static const bool [balanced](#) = false

## 12.108.1 Member Typedef Documentation

### 12.108.1.1 category

[FieldCategories::UnparametricTag category](#)

## 12.108.2 Field Documentation

### 12.108.2.1 balanced

```
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.109 FieldTraits< Givaro::ZRing< int32\_t > > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [FieldCategories::UnparametricTag category](#)

### Static Public Attributes

- static const bool [balanced](#) = false

## 12.109.1 Member Typedef Documentation

### 12.109.1.1 category

[FieldCategories::UnparametricTag category](#)

## 12.109.2 Field Documentation

### 12.109.2.1 balanced

```
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.110 FieldTraits< Givaro::ZRing< int64\_t > > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [FieldCategories::UnparametricTag category](#)

### Static Public Attributes

- static const bool [balanced](#) = false

## 12.110.1 Member Typedef Documentation

### 12.110.1.1 category

[FieldCategories::UnparametricTag category](#)

## 12.110.2 Field Documentation

### 12.110.2.1 `balanced`

```
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.111 `FieldTraits< Givaro::ZRing< Reclnt::ruint< K > > > Struct Template Reference`

```
#include <field-traits.h>
```

### Public Types

- typedef [FieldCategories::UnparametricTag](#) `category`

### Static Public Attributes

- static const bool [balanced](#) = false

## 12.111.1 Member Typedef Documentation

### 12.111.1.1 `category`

```
template<size_t K>
```

```
FieldCategories::UnparametricTag category
```

## 12.111.2 Field Documentation

### 12.111.2.1 `balanced`

```
template<size_t K>
```

```
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.112 `FieldTraits< Givaro::ZRing< uint16_t > > Struct Reference`

```
#include <field-traits.h>
```

### Public Types

- typedef [FieldCategories::UnparametricTag](#) `category`

### Static Public Attributes

- static const bool [balanced](#) = false

## 12.112.1 Member Typedef Documentation

### 12.112.1.1 `category`

```
FieldCategories::UnparametricTag category
```



### 12.112.2 Field Documentation

#### 12.112.2.1 balanced

```
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.113 FieldTraits< Givaro::ZRing< uint32\_t > > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [FieldCategories::UnparametricTag](#) category

### Static Public Attributes

- static const bool [balanced](#) = false

### 12.113.1 Member Typedef Documentation

#### 12.113.1.1 category

```
FieldCategories::UnparametricTag category
```

### 12.113.2 Field Documentation

#### 12.113.2.1 balanced

```
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.114 FieldTraits< Givaro::ZRing< uint64\_t > > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [FieldCategories::UnparametricTag](#) category

### Static Public Attributes

- static const bool [balanced](#) = false

### 12.114.1 Member Typedef Documentation

#### 12.114.1.1 category

```
FieldCategories::UnparametricTag category
```

### 12.114.2 Field Documentation

#### 12.114.2.1 balanced

```
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.115 Fixed Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.116 FixedPrecIntTag Struct Reference

Fixed precision integers above machine precision: Givaro::reclnt.

```
#include <field-traits.h>
```

### 12.116.1 Detailed Description

Fixed precision integers above machine precision: Givaro::reclnt.

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.117 ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >::FloatingPointTestDistribution Class Reference

### Public Types

- using [IntType](#) = typename make\_unsigned\_int<Element>::type

### Public Member Functions

- [FloatingPointTestDistribution](#) ()
- template<class Generator >  
Element [operator\(\)](#) (Generator &g)

### 12.117.1 Member Typedef Documentation

#### 12.117.1.1 IntType

```
template<typename Element >
using IntType = typename make_unsigned_int<Element>::type
```

### 12.117.2 Constructor & Destructor Documentation

#### 12.117.2.1 FloatingPointTestDistribution()

```
template<typename Element >
FloatingPointTestDistribution () [inline]
```

### 12.117.3 Member Function Documentation

#### 12.117.3.1 operator()()

```
template<typename Element >
template<class Generator >
Element operator\(\) (
    Generator & g) [inline]
```

The documentation for this class was generated from the following file:

- [test-simd.C](#)

## 12.118 ForStrategy1D< blocksize\_t, Cut, Param > Struct Template Reference

### Public Member Functions

- [ForStrategy1D](#) (const blocksize\_t n, const [ParSeqHelper::Parallel](#)< Cut, Param > H)
- [ForStrategy1D](#) (const blocksize\_t b, const blocksize\_t e, const [ParSeqHelper::Parallel](#)< Cut, Param > H)
- void [build](#) (const blocksize\_t n, const [ParSeqHelper::Parallel](#)< Cut, Param > H)
- blocksize\_t [initialize](#) ()
- bool [isTerminated](#) () const
- blocksize\_t [begin](#) () const
- blocksize\_t [end](#) () const
- blocksize\_t [numblocks](#) () const
- blocksize\_t [blockindex](#) () const
- blocksize\_t [operator++](#) ()

### Protected Attributes

- blocksize\_t [ibeg](#)
- blocksize\_t [iend](#)
- blocksize\_t [current](#)
- blocksize\_t [firstBlockSize](#)
- blocksize\_t [lastBlockSize](#)
- blocksize\_t [changeBS](#)
- blocksize\_t [numBlock](#)

## 12.118.1 Constructor & Destructor Documentation

### 12.118.1.1 ForStrategy1D() [1/2]

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
ForStrategy1D (
    const blocksize_t n,
    const ParSeqHelper::Parallel< Cut, Param > H) [inline]
```

### 12.118.1.2 ForStrategy1D() [2/2]

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
ForStrategy1D (
    const blocksize_t b,
    const blocksize_t e,
    const ParSeqHelper::Parallel< Cut, Param > H) [inline]
```

## 12.118.2 Member Function Documentation

### 12.118.2.1 build()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
void build (
    const blocksize_t n,
    const ParSeqHelper::Parallel< Cut, Param > H) [inline]
```

### 12.118.2.2 initialize()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t initialize () [inline]
```

### 12.118.2.3 isTerminated()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
bool isTerminated () const [inline]
```

### 12.118.2.4 begin()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t begin () const [inline]
```

### 12.118.2.5 end()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t end () const [inline]
```

### 12.118.2.6 numblocks()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t numblocks () const [inline]
```

### 12.118.2.7 blockindex()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t blockindex () const [inline]
```

### 12.118.2.8 operator++()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t operator++ () [inline]
```

## 12.118.3 Field Documentation

### 12.118.3.1 ibeg

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t ibeg [protected]
```

### 12.118.3.2 iend

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t iend [protected]
```

### 12.118.3.3 current

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t current [protected]
```

### 12.118.3.4 firstBlockSize

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t firstBlockSize [protected]
```

**12.118.3.5 lastBlockSize**

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t lastBlockSize [protected]
```

**12.118.3.6 changeBS**

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t changeBS [protected]
```

**12.118.3.7 numBlock**

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t numBlock [protected]
```

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.119 ForStrategy2D< blocksize\_t, Cut, Param > Struct Template Reference

**Public Member Functions**

- [ForStrategy2D](#) (const blocksize\_t m, const blocksize\_t n, const [ParSeqHelper::Parallel](#)< Cut, Param > H)
- blocksize\_t [initialize](#) ()
- bool [isTerminated](#) () const
- blocksize\_t [ibegin](#) () const
- blocksize\_t [jbegin](#) () const
- blocksize\_t [iend](#) () const
- blocksize\_t [jend](#) () const
- blocksize\_t [operator++](#) ()
- blocksize\_t [rownumblocks](#) () const
- blocksize\_t [colnumblocks](#) () const
- blocksize\_t [blockindex](#) () const
- blocksize\_t [rowblockindex](#) () const
- blocksize\_t [colblockindex](#) () const

**Protected Attributes**

- blocksize\_t [\\_ibeg](#)
- blocksize\_t [\\_jend](#)
- blocksize\_t [\\_jbeg](#)
- blocksize\_t [\\_jend](#)
- blocksize\_t [rowBlockSize](#)
- blocksize\_t [colBlockSize](#)
- blocksize\_t [current](#)
- blocksize\_t [lastRBS](#)
- blocksize\_t [lastCBS](#)
- blocksize\_t [changeRBS](#)
- blocksize\_t [changeCBS](#)
- blocksize\_t [numRowBlock](#)
- blocksize\_t [numColBlock](#)
- blocksize\_t [BLOCKS](#)

## Friends

- `std::ostream & operator<< (std::ostream &out, const ForStrategy2D &FS2D)`

## 12.119.1 Constructor & Destructor Documentation

### 12.119.1.1 ForStrategy2D()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
ForStrategy2D (
    const blocksize_t m,
    const blocksize_t n,
    const ParSeqHelper::Parallel< Cut, Param > H) [inline]
```

## 12.119.2 Member Function Documentation

### 12.119.2.1 initialize()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t initialize () [inline]
```

### 12.119.2.2 isTerminated()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
bool isTerminated () const [inline]
```

### 12.119.2.3 ibegin()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t ibegin () const [inline]
```

### 12.119.2.4 jbegin()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t jbegin () const [inline]
```

### 12.119.2.5 iend()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t iend () const [inline]
```

### 12.119.2.6 jend()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t jend () const [inline]
```

### 12.119.2.7 operator++()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t operator++ () [inline]
```

**12.119.2.8 rownumblocks()**

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t rownumblocks () const [inline]
```

**12.119.2.9 colnumblocks()**

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t colnumblocks () const [inline]
```

**12.119.2.10 blockindex()**

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t blockindex () const [inline]
```

**12.119.2.11 rowblockindex()**

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t rowblockindex () const [inline]
```

**12.119.2.12 colblockindex()**

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t colblockindex () const [inline]
```

**12.119.3 Friends And Related Symbol Documentation****12.119.3.1 operator<<**

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
std::ostream & operator<< (
    std::ostream & out,
    const ForStrategy2D< blocksize_t, Cut, Param > & FS2D) [friend]
```

**12.119.4 Field Documentation****12.119.4.1 \_ibeg**

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t _ibeg [protected]
```

**12.119.4.2 \_iend**

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t _iend [protected]
```

**12.119.4.3 \_jbeg**

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t _jbeg [protected]
```

**12.119.4.4 \_jend**

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t _jend [protected]
```

**12.119.4.5 rowBlockSize**

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t rowBlockSize [protected]
```

**12.119.4.6 colBlockSize**

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t colBlockSize [protected]
```

**12.119.4.7 current**

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t current [protected]
```

**12.119.4.8 lastRBS**

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t lastRBS [protected]
```

**12.119.4.9 lastCBS**

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t lastCBS [protected]
```

**12.119.4.10 changeRBS**

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t changeRBS [protected]
```

**12.119.4.11 changeCBS**

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t changeCBS [protected]
```

**12.119.4.12 numRowsBlock**

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t numRowsBlock [protected]
```

**12.119.4.13 numColBlock**

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t numColBlock [protected]
```



#### 12.119.4.14 BLOCKS

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
```

```
blocksize_t BLOCKS [protected]
```

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

### 12.120 ftrmmLeftLowerNoTransNonUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

### 12.121 ftrmmLeftLowerNoTransUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

### 12.122 ftrmmLeftLowerTransNonUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

### 12.123 ftrmmLeftLowerTransUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

### 12.124 ftrmmLeftUpperNoTransNonUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

### 12.125 ftrmmLeftUpperNoTransUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

### 12.126 ftrmmLeftUpperTransNonUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.127 `ftrmmLeftUpperTransUnit< Element >` Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.128 `ftrmmRightLowerNoTransNonUnit< Element >` Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.129 `ftrmmRightLowerNoTransUnit< Element >` Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.130 `ftrmmRightLowerTransNonUnit< Element >` Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.131 `ftrmmRightLowerTransUnit< Element >` Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.132 `ftrmmRightUpperNoTransNonUnit< Element >` Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.133 `ftrmmRightUpperNoTransUnit< Element >` Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.134 `ftrmmRightUpperTransNonUnit< Element >` Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.135 ftrmmRightUpperTransUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.136 ftrsmLeftLowerNoTransNonUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.137 ftrsmLeftLowerNoTransUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.138 ftrsmLeftLowerTransNonUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.139 ftrsmLeftLowerTransUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.140 ftrsmLeftUpperNoTransNonUnit< Element > Class Template Reference

Computes the maximal size for delaying the modular reduction in a triangular system resolution.

### 12.140.1 Detailed Description

**template<class Element>**

**class FFLAS::Protected::ftrsmLeftUpperNoTransNonUnit< Element >**

Computes the maximal size for delaying the modular reduction in a triangular system resolution.

Compute the maximal dimension  $k$ , such that a unit diagonal triangular system of dimension  $k$  can be solved over  $Z$  without overflow of the underlying floating point representation.

**Bibliography** • Dumas, Giorgi, Pernet 06, arXiv:cs/0601133.

Parameters

|     |                                      |
|-----|--------------------------------------|
| $F$ | Finite Field/Ring of the computation |
|-----|--------------------------------------|

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

### 12.141 **ftrsmLeftUpperNoTransUnit**< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

### 12.142 **ftrsmLeftUpperTransNonUnit**< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

### 12.143 **ftrsmLeftUpperTransUnit**< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

### 12.144 **ftrsmRightLowerNoTransNonUnit**< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

### 12.145 **ftrsmRightLowerNoTransUnit**< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

### 12.146 **ftrsmRightLowerTransNonUnit**< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

### 12.147 **ftrsmRightLowerTransUnit**< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

### 12.148 **ftrsmRightUpperNoTransNonUnit**< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.149 ftrsmRightUpperNoTransUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.150 ftrsmRightUpperTransNonUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.151 ftrsmRightUpperTransUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.152 GenericTag Struct Reference

default is generic

```
#include <field-traits.h>
```

### 12.152.1 Detailed Description

default is generic

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.153 GenericTag Struct Reference

generic ring.

```
#include <field-traits.h>
```

### 12.153.1 Detailed Description

generic ring.

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.154 Grain Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.155 has\_minus\_eq\_impl< C > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

**Static Public Attributes**

- static constexpr bool [value](#) = type::value

**12.155.1 Field Documentation****12.155.1.1 value**

```
template<typename C >
bool value = type::value [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

**12.156 has\_minus\_impl< C > Struct Template Reference**

```
#include <sparse_matrix_traits.h>
```

**Static Public Attributes**

- static constexpr bool [value](#) = type::value

**12.156.1 Field Documentation****12.156.1.1 value**

```
template<typename C >
bool value = type::value [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

**12.157 has\_mul\_eq\_impl< C > Struct Template Reference**

```
#include <sparse_matrix_traits.h>
```

**Static Public Attributes**

- static constexpr bool [value](#) = type::value

**12.157.1 Field Documentation****12.157.1.1 value**

```
template<typename C >
bool value = type::value [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

**12.158 has\_mul\_impl< C > Struct Template Reference**

```
#include <sparse_matrix_traits.h>
```

**Static Public Attributes**

- static constexpr bool [value](#) = type::value

### 12.158.1 Field Documentation

#### 12.158.1.1 `value`

```
template<typename C >
bool value = type::value [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.159 `has_operation< T >` Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

### Static Public Attributes

- static constexpr bool `value`

### 12.159.1 Field Documentation

#### 12.159.1.1 `value`

```
template<class T >
bool value [static], [constexpr]
Initial value:
= (has_plus<T>::value && has_minus<T>::value && has_equal<T>::value &&
    has_plus_eq<T>::value && has_minus_eq<T>::value && has_mul<T>::value
    && has_mul_eq<T>::value)
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.160 `has_plus_eq_impl< C >` Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

### Static Public Attributes

- static constexpr bool `value` = type::value

### 12.160.1 Field Documentation

#### 12.160.1.1 `value`

```
template<typename C >
bool value = type::value [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.161 `has_plus_impl< C >` Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

### Static Public Attributes

- static constexpr bool `value` = type::value

### 12.161.1 Field Documentation

#### 12.161.1.1 value

```
template<typename C >
bool value = type::value [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.162 HelperFlag Struct Reference

```
#include <fflas_sparse.h>
```

### Static Public Attributes

- static constexpr uint64\_t [none](#) = 0\_ui64
- static constexpr uint64\_t [coo](#) = 1\_ui64
- static constexpr uint64\_t [csr](#) = 1\_ui64 << 1
- static constexpr uint64\_t [ell](#) = 1\_ui64 << 2
- static constexpr uint64\_t [aut](#) = 1\_ui64 << 32
- static constexpr uint64\_t [pm1](#) = 1\_ui64 << 33

### 12.162.1 Field Documentation

#### 12.162.1.1 none

```
uint64_t none = 0_ui64 [static], [constexpr]
```

#### 12.162.1.2 coo

```
uint64_t coo = 1_ui64 [static], [constexpr]
```

#### 12.162.1.3 csr

```
uint64_t csr = 1_ui64 << 1 [static], [constexpr]
```

#### 12.162.1.4 ell

```
uint64_t ell = 1_ui64 << 2 [static], [constexpr]
```

#### 12.162.1.5 aut

```
uint64_t aut = 1_ui64 << 32 [static], [constexpr]
```

#### 12.162.1.6 pm1

```
uint64_t pm1 = 1_ui64 << 33 [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [fflas\\_sparse.h](#)

## 12.163 HelperMod< Field, ElementTraits > Struct Template Reference

The documentation for this struct was generated from the following file:

- [fflas\\_freduce.inl](#)



## 12.164 HelperMod< Field, ElementCategories::MachineIntTag > Struct Template Reference

### Public Member Functions

- [HelperMod](#) ()
- [HelperMod](#) (const [Field](#) &F)

### Data Fields

- [Field::Element](#) p
- double [invp](#)
- double [min](#)
- double [max](#)
- int64\_t [pow50rem](#)

### 12.164.1 Constructor & Destructor Documentation

#### 12.164.1.1 HelperMod() [1/2]

```
template<class Field >  
HelperMod () [inline]
```

#### 12.164.1.2 HelperMod() [2/2]

```
template<class Field >  
HelperMod (  
    const Field & F) [inline]
```

### 12.164.2 Field Documentation

#### 12.164.2.1 p

```
template<class Field >  
Field::Element p
```

#### 12.164.2.2 invp

```
template<class Field >  
double invp
```

#### 12.164.2.3 min

```
template<class Field >  
double min
```

#### 12.164.2.4 max

```
template<class Field >  
double max
```

#### 12.164.2.5 pow50rem

```
template<class Field >  
int64_t pow50rem
```

The documentation for this struct was generated from the following file:

- [fflas\\_freduce.inl](#)

## 12.165 HelperMod< Field, FFLAS::ElementCategories::ArbitraryPrecIntTag > Struct Template Reference

### Public Member Functions

- [HelperMod \(\)](#)
- [HelperMod \(const \[Field\]\(#\) &F\)](#)

### Data Fields

- [Field::Element p](#)

### 12.165.1 Constructor & Destructor Documentation

#### 12.165.1.1 HelperMod() [1/2]

```
template<class Field >
HelperMod () [inline]
```

#### 12.165.1.2 HelperMod() [2/2]

```
template<class Field >
HelperMod (
    const Field & F) [inline]
```

### 12.165.2 Field Documentation

#### 12.165.2.1 p

```
template<class Field >
Field::Element p
```

The documentation for this struct was generated from the following file:

- [fflas\\_freduce.inl](#)

## 12.166 HelperMod< Field, FFLAS::ElementCategories::FixedPrecIntTag > Struct Template Reference

### Public Member Functions

- [HelperMod \(\)](#)
- [HelperMod \(const \[Field\]\(#\) &F\)](#)

### Data Fields

- [Field::Element p](#)

### 12.166.1 Constructor & Destructor Documentation

#### 12.166.1.1 HelperMod() [1/2]

```
template<class Field >
HelperMod () [inline]
```

#### 12.166.1.2 HelperMod() [2/2]

```
template<class Field >
HelperMod (
    const Field & F) [inline]
```

## 12.166.2 Field Documentation

### 12.166.2.1 p

```
template<class Field >
Field::Element p
```

The documentation for this struct was generated from the following file:

- [fflas\\_freduce.inl](#)

## 12.167 HelperMod< Field, FFLAS::ElementCategories::MachineFloatTag > Struct Template Reference

### Public Member Functions

- [HelperMod](#) ()
- [HelperMod](#) (const [Field](#) &F)

### Data Fields

- [Field::Element](#) p
- [Field::Element](#) invp
- [Field::Element](#) min
- [Field::Element](#) max

## 12.167.1 Constructor & Destructor Documentation

### 12.167.1.1 HelperMod() [1/2]

```
template<class Field >
HelperMod () [inline]
```

### 12.167.1.2 HelperMod() [2/2]

```
template<class Field >
HelperMod (
    const Field & F) [inline]
```

## 12.167.2 Field Documentation

### 12.167.2.1 p

```
template<class Field >
Field::Element p
```

### 12.167.2.2 invp

```
template<class Field >
Field::Element invp
```

### 12.167.2.3 min

```
template<class Field >
Field::Element min
```

### 12.167.2.4 max

```
template<class Field >
Field::Element max
```

The documentation for this struct was generated from the following file:

- [fflas\\_freduce.inl](#)

## 12.168 Hybrid Struct Reference

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

## 12.169 Info Struct Reference

### Public Member Functions

- [Info](#) (uint64\_t it, uint64\_t s, uint64\_t p)
- [Info](#) ()=default
- [Info](#) (const [Info](#) &)=default
- [Info](#) ([Info](#) &&)=default
- [Info](#) & [operator=](#) (const [Info](#) &)=default
- [Info](#) & [operator=](#) ([Info](#) &&)=default

### Data Fields

- uint64\_t [size](#) = 0
- uint64\_t [perm](#) = 0
- uint64\_t [begin](#) = 0

### 12.169.1 Constructor & Destructor Documentation

#### 12.169.1.1 Info() [1/4]

```
Info (
    uint64_t it,
    uint64_t s,
    uint64_t p) [inline]
```

#### 12.169.1.2 Info() [2/4]

```
Info () [default]
```

#### 12.169.1.3 Info() [3/4]

```
Info (
    const Info & ) [default]
```

#### 12.169.1.4 Info() [4/4]

```
Info (
    Info && ) [default]
```

### 12.169.2 Member Function Documentation

#### 12.169.2.1 operator=() [1/2]

```
Info & operator= (
    const Info & ) [default]
```

#### 12.169.2.2 operator=() [2/2]

```
Info & operator= (
    Info && ) [default]
```

### 12.169.3 Field Documentation

#### 12.169.3.1 size

`uint64_t size = 0`

#### 12.169.3.2 perm

`uint64_t perm = 0`

#### 12.169.3.3 begin

`uint64_t begin = 0`

The documentation for this struct was generated from the following file:

- [csr\\_hyb\\_utils.inl](#)

## 12.170 Info Struct Reference

### Public Member Functions

- [Info](#) (`uint64_t it`, `uint64_t s`, `uint64_t p`)
- [Info](#) ()=default
- [Info](#) (`const Info &`)=default
- [Info](#) (`Info &&`)=default
- [Info & operator=](#) (`const Info &`)=default
- [Info & operator=](#) (`Info &&`)=default

### Data Fields

- `uint64_t` [size](#) = 0
- `uint64_t` [perm](#) = 0
- `uint64_t` [begin](#) = 0

### 12.170.1 Constructor & Destructor Documentation

#### 12.170.1.1 Info() [1/4]

```
Info (
    uint64_t it,
    uint64_t s,
    uint64_t p) [inline]
```

#### 12.170.1.2 Info() [2/4]

```
Info () [default]
```

#### 12.170.1.3 Info() [3/4]

```
Info (
    const Info & ) [default]
```

#### 12.170.1.4 Info() [4/4]

```
Info (
    Info && ) [default]
```

## 12.170.2 Member Function Documentation

### 12.170.2.1 operator=() [1/2]

```
Info & operator= (
    const Info & ) [default]
```

### 12.170.2.2 operator=() [2/2]

```
Info & operator= (
    Info && ) [default]
```

## 12.170.3 Field Documentation

### 12.170.3.1 size

```
uint64_t size = 0
```

### 12.170.3.2 perm

```
uint64_t perm = 0
```

### 12.170.3.3 begin

```
uint64_t begin = 0
```

The documentation for this struct was generated from the following file:

- [sell\\_utils.inl](#)

## 12.171 is\_all\_same< Args > Struct Template Reference

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.172 is\_all\_same< T, Args... > Struct Template Reference

### Static Public Attributes

- static constexpr bool [value](#) = [ALL](#)<std::is\_same<typename decay<T>::type, typename decay<Args>::type>::value...>::value

### 12.172.1 Field Documentation

#### 12.172.1.1 value

```
template<typename T , typename... Args>
bool value = ALL<std::is_same<typename decay<T>::type, typename decay<Args>::type>::value...>::value [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.173 is\_all\_same<> Struct Reference

### Static Public Attributes

- static constexpr bool [value](#) = true

### 12.173.1 Field Documentation

#### 12.173.1.1 value

```
bool value = true [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.174 is\_simd< T > Struct Template Reference

```
#include <fflas_simd.h>
```

### Public Types

- using [type](#) = std::integral\_constant<bool, false>

### Static Public Attributes

- static const constexpr bool [value](#) = false

### 12.174.1 Member Typedef Documentation

#### 12.174.1.1 type

```
template<class T >
using type = std::integral_constant<bool, false>
```

### 12.174.2 Field Documentation

#### 12.174.2.1 value

```
template<class T >
const constexpr bool value = false [static], [constexpr]
```

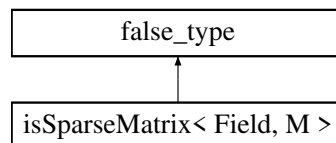
The documentation for this struct was generated from the following file:

- [fflas\\_simd.h](#)

## 12.175 isSparseMatrix< Field, M > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isSparseMatrix< Field, M >:



The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.176 isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::COO > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::COO > >:



The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.177 isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::COO\_ZO > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::COO\_ZO > >:



The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.178 isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::CSR > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::CSR > >:



The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.179 isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::CSR\_HYB > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::CSR\_HYB > >:





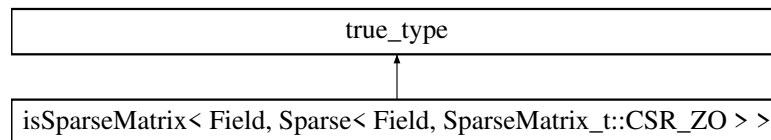
The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.180 isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::CSR\_ZO > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::CSR\_ZO > >:



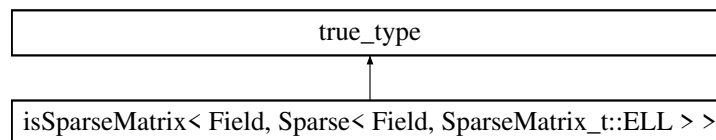
The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.181 isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL > >:



The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.182 isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL\_simd > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL\_simd > >:



The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

### 12.183 `isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > >` Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for `isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > >`:



The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

### 12.184 `isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_ZO > >` Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for `isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_ZO > >`:



The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

### 12.185 `isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::HYB_ZO > >` Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for `isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::HYB_ZO > >`:



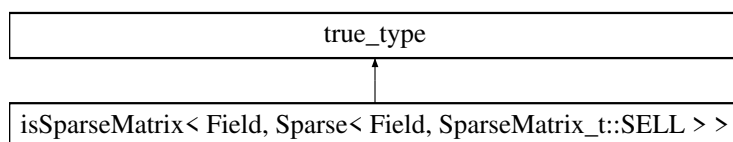
The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

### 12.186 `isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL > >` Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for `isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL > >`:



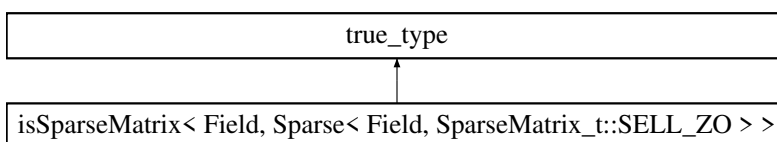
The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.187 isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::SELL\_ZO > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::SELL\_ZO > >:



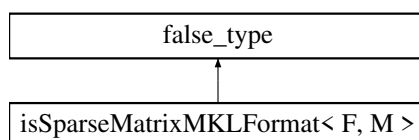
The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.188 isSparseMatrixMKLFormat< F, M > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isSparseMatrixMKLFormat< F, M >:



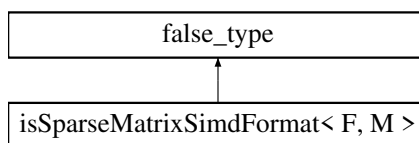
The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.189 isSparseMatrixSimdFormat< F, M > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isSparseMatrixSimdFormat< F, M >:



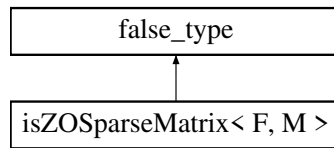
The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.190 isZOSparseMatrix< F, M > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isZOSparseMatrix< F, M >:



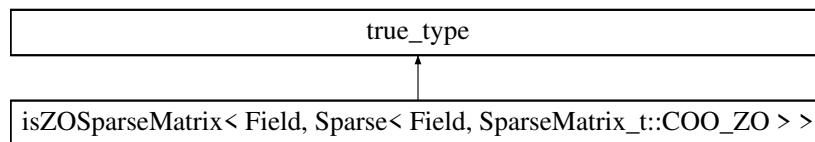
The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.191 isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::COO\_ZO > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::COO\_ZO > >:



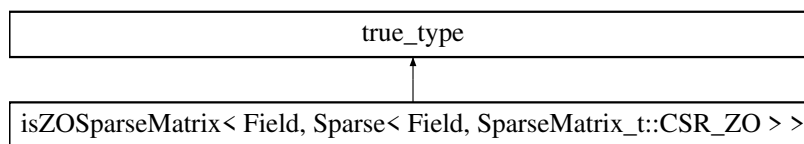
The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.192 isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::CSR\_ZO > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::CSR\_ZO > >:



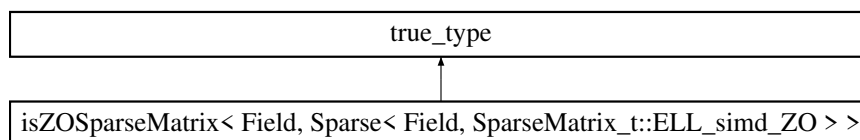
The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.193 isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL\_simd\_ZO > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL\_simd\_ZO > >:



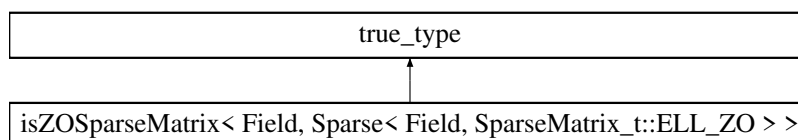
The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.194 isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL\_ZO > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL\_ZO > >:



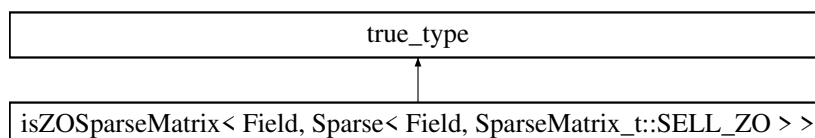
The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.195 isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::SELL\_ZO > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::SELL\_ZO > >:



The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.196 Iterative Struct Reference

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

## 12.197 LazyTag Struct Reference

Performs field operations with delayed mod only when necessary. Result may not be reduced.

```
#include <field-traits.h>
```

### 12.197.1 Detailed Description

Performs field operations with delayed mod only when necessary. Result may not be reduced.  
The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.198 `limits< T >` Struct Template Reference

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.199 `limits< char >` Struct Reference

```
#include <flimits.h>
```

### Public Types

- typedef char [T](#)

### Static Public Member Functions

- static constexpr char [max](#) () noexcept
- static constexpr char [min](#) () noexcept
- static constexpr int32\_t [digits](#) () noexcept

### 12.199.1 Member Typedef Documentation

#### 12.199.1.1 `T`

char [T](#)

### 12.199.2 Member Function Documentation

#### 12.199.2.1 `max()`

```
static constexpr char max () [inline], [static], [constexpr], [noexcept]
```

#### 12.199.2.2 `min()`

```
static constexpr char min () [inline], [static], [constexpr], [noexcept]
```

#### 12.199.2.3 `digits()`

```
static constexpr int32_t digits () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.200 `limits< double >` Struct Reference

```
#include <flimits.h>
```

### Public Types

- typedef double [T](#)

### Static Public Member Functions

- static constexpr int64\_t [max](#) () noexcept
- static constexpr int64\_t [min](#) () noexcept
- static constexpr int32\_t [digits](#) () noexcept

## 12.200.1 Member Typedef Documentation

### 12.200.1.1 T

double [T](#)

## 12.200.2 Member Function Documentation

### 12.200.2.1 max()

```
static constexpr int64_t max () [inline], [static], [constexpr], [noexcept]
```

### 12.200.2.2 min()

```
static constexpr int64_t min () [inline], [static], [constexpr], [noexcept]
```

### 12.200.2.3 digits()

```
static constexpr int32_t digits () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.201 limits< float > Struct Reference

```
#include <flimits.h>
```

### Public Types

- typedef float [T](#)

### Static Public Member Functions

- static constexpr int32\_t [max](#) () noexcept
- static constexpr int32\_t [min](#) () noexcept
- static constexpr int32\_t [digits](#) () noexcept

## 12.201.1 Member Typedef Documentation

### 12.201.1.1 T

float [T](#)

## 12.201.2 Member Function Documentation

### 12.201.2.1 max()

```
static constexpr int32_t max () [inline], [static], [constexpr], [noexcept]
```

### 12.201.2.2 min()

```
static constexpr int32_t min () [inline], [static], [constexpr], [noexcept]
```

### 12.201.2.3 digits()

```
static constexpr int32_t digits () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.202 limits< Givaro::Integer > Struct Reference

```
#include <flimits.h>
```

### Public Types

- typedef Givaro::Integer [T](#)

### Static Public Member Functions

- static constexpr int [max](#) () noexcept
- static constexpr int [min](#) () noexcept

### 12.202.1 Member Typedef Documentation

#### 12.202.1.1 T

```
Givaro::Integer T
```

### 12.202.2 Member Function Documentation

#### 12.202.2.1 max()

```
static constexpr int max () [inline], [static], [constexpr], [noexcept]
```

#### 12.202.2.2 min()

```
static constexpr int min () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.203 limits< int > Struct Reference

```
#include <flimits.h>
```

### Public Types

- typedef int [T](#)

### Static Public Member Functions

- static constexpr int [max](#) () noexcept
- static constexpr int [min](#) () noexcept
- static constexpr int32\_t [digits](#) () noexcept

### 12.203.1 Member Typedef Documentation

#### 12.203.1.1 T

```
int T
```



## 12.203.2 Member Function Documentation

### 12.203.2.1 max()

```
static constexpr int max () [inline], [static], [constexpr], [noexcept]
```

### 12.203.2.2 min()

```
static constexpr int min () [inline], [static], [constexpr], [noexcept]
```

### 12.203.2.3 digits()

```
static constexpr int32_t digits () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.204 limits< long > Struct Reference

```
#include <flimits.h>
```

### Public Types

- typedef long [T](#)

### Static Public Member Functions

- static constexpr long [max](#) () noexcept
- static constexpr long [min](#) () noexcept
- static constexpr int32\_t [digits](#) () noexcept

## 12.204.1 Member Typedef Documentation

### 12.204.1.1 T

```
long T
```

## 12.204.2 Member Function Documentation

### 12.204.2.1 max()

```
static constexpr long max () [inline], [static], [constexpr], [noexcept]
```

### 12.204.2.2 min()

```
static constexpr long min () [inline], [static], [constexpr], [noexcept]
```

### 12.204.2.3 digits()

```
static constexpr int32_t digits () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.205 limits< long long > Struct Reference

```
#include <flimits.h>
```

### Public Types

- typedef long long [T](#)

### Static Public Member Functions

- static constexpr long long [max](#) () noexcept
- static constexpr long long [min](#) () noexcept
- static constexpr int32\_t [digits](#) () noexcept

## 12.205.1 Member Typedef Documentation

### 12.205.1.1 T

long long [T](#)

## 12.205.2 Member Function Documentation

### 12.205.2.1 max()

static constexpr long long max () [inline], [static], [constexpr], [noexcept]

### 12.205.2.2 min()

static constexpr long long min () [inline], [static], [constexpr], [noexcept]

### 12.205.2.3 digits()

static constexpr int32\_t digits () [inline], [static], [constexpr], [noexcept]

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.206 limits< RecInt::rint< K > > Struct Template Reference

```
#include <flimits.h>
```

### Public Types

- typedef [RecInt::ruint](#)< K > [T](#)

### Static Public Member Functions

- static constexpr [RecInt::rint](#)< K > [max](#) () noexcept
- static constexpr [RecInt::rint](#)< K > [min](#) () noexcept

## 12.206.1 Member Typedef Documentation

### 12.206.1.1 T

template<size\_t K>  
[RecInt::ruint](#)<K> [T](#)

## 12.206.2 Member Function Documentation

### 12.206.2.1 max()

template<size\_t K>  
static constexpr [RecInt::rint](#)< K > max () [inline], [static], [constexpr], [noexcept]

**12.206.2.2 min()**

```
template<size_t K>
static constexpr RecInt::ruint< K > min () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

**12.207 limits< RecInt::ruint< K > > Struct Template Reference**

```
#include <flimits.h>
```

**Public Types**

- typedef [RecInt::ruint< K > T](#)

**Static Public Member Functions**

- static constexpr [RecInt::ruint< K > max](#) () noexcept
- static constexpr [RecInt::ruint< K > min](#) () noexcept

**12.207.1 Member Typedef Documentation****12.207.1.1 T**

```
template<size_t K>
RecInt::ruint<K> T
```

**12.207.2 Member Function Documentation****12.207.2.1 max()**

```
template<size_t K>
static constexpr RecInt::ruint< K > max () [inline], [static], [constexpr], [noexcept]
```

**12.207.2.2 min()**

```
template<size_t K>
static constexpr RecInt::ruint< K > min () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

**12.208 limits< short int > Struct Reference**

```
#include <flimits.h>
```

**Public Types**

- typedef short int [T](#)

**Static Public Member Functions**

- static constexpr short int [max](#) () noexcept
- static constexpr short int [min](#) () noexcept
- static constexpr int32\_t [digits](#) () noexcept

## 12.208.1 Member Typedef Documentation

### 12.208.1.1 T

short int [T](#)

## 12.208.2 Member Function Documentation

### 12.208.2.1 max()

static constexpr short int max () [inline], [static], [constexpr], [noexcept]

### 12.208.2.2 min()

static constexpr short int min () [inline], [static], [constexpr], [noexcept]

### 12.208.2.3 digits()

static constexpr int32\_t digits () [inline], [static], [constexpr], [noexcept]

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.209 limits< signed char > Struct Reference

```
#include <flimits.h>
```

### Public Types

- typedef signed char [T](#)

### Static Public Member Functions

- static constexpr signed char [max](#) () noexcept
- static constexpr signed char [min](#) () noexcept
- static constexpr int32\_t [digits](#) () noexcept

## 12.209.1 Member Typedef Documentation

### 12.209.1.1 T

signed char [T](#)

## 12.209.2 Member Function Documentation

### 12.209.2.1 max()

static constexpr signed char max () [inline], [static], [constexpr], [noexcept]

### 12.209.2.2 min()

static constexpr signed char min () [inline], [static], [constexpr], [noexcept]

### 12.209.2.3 digits()

static constexpr int32\_t digits () [inline], [static], [constexpr], [noexcept]

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.210 limits< unsigned char > Struct Reference

```
#include <flimits.h>
```

### Public Types

- typedef unsigned char [T](#)

### Static Public Member Functions

- static constexpr unsigned char [max](#) () noexcept
- static constexpr unsigned char [min](#) () noexcept
- static constexpr int32\_t [digits](#) () noexcept

### 12.210.1 Member Typedef Documentation

#### 12.210.1.1 T

```
unsigned char T
```

### 12.210.2 Member Function Documentation

#### 12.210.2.1 max()

```
static constexpr unsigned char max () [inline], [static], [constexpr], [noexcept]
```

#### 12.210.2.2 min()

```
static constexpr unsigned char min () [inline], [static], [constexpr], [noexcept]
```

#### 12.210.2.3 digits()

```
static constexpr int32_t digits () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.211 limits< unsigned int > Struct Reference

```
#include <flimits.h>
```

### Public Types

- typedef unsigned int [T](#)

### Static Public Member Functions

- static constexpr unsigned int [max](#) () noexcept
- static constexpr unsigned int [min](#) () noexcept
- static constexpr int32\_t [digits](#) () noexcept

### 12.211.1 Member Typedef Documentation

#### 12.211.1.1 T

```
unsigned int T
```

## 12.211.2 Member Function Documentation

### 12.211.2.1 max()

```
static constexpr unsigned int max () [inline], [static], [constexpr], [noexcept]
```

### 12.211.2.2 min()

```
static constexpr unsigned int min () [inline], [static], [constexpr], [noexcept]
```

### 12.211.2.3 digits()

```
static constexpr int32_t digits () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.212 limits< unsigned long > Struct Reference

```
#include <flimits.h>
```

### Public Types

- typedef unsigned long [T](#)

### Static Public Member Functions

- static constexpr unsigned long [max](#) () noexcept
- static constexpr unsigned long [min](#) () noexcept
- static constexpr int32\_t [digits](#) () noexcept

## 12.212.1 Member Typedef Documentation

### 12.212.1.1 T

```
unsigned long T
```

## 12.212.2 Member Function Documentation

### 12.212.2.1 max()

```
static constexpr unsigned long max () [inline], [static], [constexpr], [noexcept]
```

### 12.212.2.2 min()

```
static constexpr unsigned long min () [inline], [static], [constexpr], [noexcept]
```

### 12.212.2.3 digits()

```
static constexpr int32_t digits () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.213 limits< unsigned long long > Struct Reference

```
#include <flimits.h>
```

### Public Types

- typedef unsigned long long [T](#)

### Static Public Member Functions

- static constexpr unsigned long long [max](#) () noexcept
- static constexpr unsigned long long [min](#) () noexcept
- static constexpr int32\_t [digits](#) () noexcept

## 12.213.1 Member Typedef Documentation

### 12.213.1.1 T

unsigned long long [T](#)

## 12.213.2 Member Function Documentation

### 12.213.2.1 max()

static constexpr unsigned long long max () [inline], [static], [constexpr], [noexcept]

### 12.213.2.2 min()

static constexpr unsigned long long min () [inline], [static], [constexpr], [noexcept]

### 12.213.2.3 digits()

static constexpr int32\_t digits () [inline], [static], [constexpr], [noexcept]

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.214 limits< unsigned short int > Struct Reference

```
#include <flimits.h>
```

### Public Types

- typedef unsigned short int [T](#)

### Static Public Member Functions

- static constexpr unsigned short int [max](#) () noexcept
- static constexpr unsigned short int [min](#) () noexcept
- static constexpr int32\_t [digits](#) () noexcept

## 12.214.1 Member Typedef Documentation

### 12.214.1.1 T

unsigned short int [T](#)

## 12.214.2 Member Function Documentation

### 12.214.2.1 max()

static constexpr unsigned short int max () [inline], [static], [constexpr], [noexcept]

### 12.214.2.2 min()

static constexpr unsigned short int min () [inline], [static], [constexpr], [noexcept]

### 12.214.2.3 digits()

```
static constexpr int32_t digits () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.215 MachineFloatTag Struct Reference

float or double

```
#include <field-traits.h>
```

### 12.215.1 Detailed Description

float or double

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.216 MachineIntTag Struct Reference

short, int, long, long long, and unsigned variants

```
#include <field-traits.h>
```

### 12.216.1 Detailed Description

short, int, long, long long, and unsigned variants

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.217 MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait > Struct Template Reference

### Public Types

- typedef [MMHelper](#)< [Field](#), [AlgoTrait](#), [ModeTrait](#), [ParSeqTrait](#) > [Self\\_t](#)
- typedef [associatedDelayedField](#)< [constField](#) >::type [DelayedField\\_t](#)
- typedef [associatedDelayedField](#)< [constField](#) >::field [DelayedField](#)
- typedef [DelayedField](#)::Element [DFElt](#)

### Public Member Functions

- void [initC](#) ()
- void [initA](#) ()
- void [initB](#) ()
- void [initOut](#) ()
- [size\\_t](#) [MaxDelayedDim](#) ([DFElt](#) beta)
- bool [Aunfit](#) ()
- bool [Bunfit](#) ()
- void [setOutBounds](#) (const [size\\_t](#) k, const [DFElt](#) alpha, const [DFElt](#) beta)
- bool [checkA](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_TRANSPOSE](#) ta, const [size\\_t](#) M, const [size\\_t](#) N, typename [Field::ConstElement\\_ptr](#) A, const [size\\_t](#) lda)
- bool [checkB](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_TRANSPOSE](#) tb, const [size\\_t](#) M, const [size\\_t](#) N, typename [Field::ConstElement\\_ptr](#) B, const [size\\_t](#) ldb)
- bool [checkOut](#) (const [Field](#) &F, const [size\\_t](#) M, const [size\\_t](#) N, typename [Field::ConstElement\\_ptr](#) A, const [size\\_t](#) lda)



- bool [checkOut](#) (const [Field](#) &F, [FFLAS\\_UPLO](#) uplo, const size\_t M, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda)
- [MMHelper](#) ()
- [MMHelper](#) (const [Field](#) &F, size\_t m, size\_t k, size\_t n, ParSeqTrait \_PS)
- [MMHelper](#) (const [Field](#) &F, int w, ParSeqTrait \_PS=ParSeqTrait())
- template<class F2 , typename AlgoT2 , typename FT2 , typename PS2 >  
[MMHelper](#) ([MMHelper](#)< F2, AlgoT2, FT2, PS2 > &WH)
- [MMHelper](#) (const [Field](#) &F, int w, [DFElt](#) \_Amin, [DFElt](#) \_Amax, [DFElt](#) \_Bmin, [DFElt](#) \_Bmax, [DFElt](#) \_Cmin, [DFElt](#) \_Cmax, ParSeqTrait \_PS=ParSeqTrait())

## Data Fields

- int [recLevel](#)
- [DFElt](#) [FieldMin](#)
- [DFElt](#) [FieldMax](#)
- [DFElt](#) [Amin](#)
- [DFElt](#) [Amax](#)
- [DFElt](#) [Bmin](#)
- [DFElt](#) [Bmax](#)
- [DFElt](#) [Cmin](#)
- [DFElt](#) [Cmax](#)
- [DFElt](#) [Outmin](#)
- [DFElt](#) [Outmax](#)
- [DFElt](#) [MaxStorableValue](#)
- const [DelayedField\\_t](#) [delayedField](#)
- ParSeqTrait [parseq](#)

## Friends

- std::ostream & [operator<<](#) (std::ostream &out, const [Self\\_t](#) &M)

## 12.217.1 Member Typedef Documentation

### 12.217.1.1 Self\_t

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
MMHelper<Field,AlgoTrait,ModeTrait,ParSeqTrait> Self\_t
```

### 12.217.1.2 DelayedField\_t

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
associatedDelayedField<constField>::type DelayedField\_t
```

### 12.217.1.3 DelayedField

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
associatedDelayedField<constField>::field DelayedField
```

### 12.217.1.4 DFElt

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
DelayedField::Element DFElt
```

## 12.217.2 Constructor & Destructor Documentation

### 12.217.2.1 MMHelper() [1/5]

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
MMHelper () [inline]
```

**12.217.2.2 MMHelper() [2/5]**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
MMHelper (
    const Field & F,
    size_t m,
    size_t k,
    size_t n,
    ParSeqTrait _PS) [inline]
```

**12.217.2.3 MMHelper() [3/5]**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
MMHelper (
    const Field & F,
    int w,
    ParSeqTrait _PS = ParSeqTrait()) [inline]
```

**12.217.2.4 MMHelper() [4/5]**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
template<class F2 , typename AlgoT2 , typename FT2 , typename PS2 >
MMHelper (
    MMHelper< F2, AlgoT2, FT2, PS2 > & WH) [inline]
```

**12.217.2.5 MMHelper() [5/5]**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
MMHelper (
    const Field & F,
    int w,
    DFelt _Amin,
    DFelt _Amax,
    DFelt _Bmin,
    DFelt _Bmax,
    DFelt _Cmin,
    DFelt _Cmax,
    ParSeqTrait _PS = ParSeqTrait()) [inline]
```

**12.217.3 Member Function Documentation****12.217.3.1 initC()**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
void initC () [inline]
```

**12.217.3.2 initA()**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
void initA () [inline]
```

**12.217.3.3 initB()**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
void initB () [inline]
```

**12.217.3.4 initOut()**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
void initOut () [inline]
```

**12.217.3.5 MaxDelayedDim()**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
size_t MaxDelayedDim (
    DFelt beta) [inline]
```

**12.217.3.6 Aunfit()**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
bool Aunfit () [inline]
```

**12.217.3.7 Bunfit()**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
bool Bunfit () [inline]
```

**12.217.3.8 setOutBounds()**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
void setOutBounds (
    const size_t k,
    const DFelt alpha,
    const DFelt beta) [inline]
```

**12.217.3.9 checkA()**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
bool checkA (
    const Field & F,
    const FFLAS::FFLAS_TRANSPOSE ta,
    const size_t M,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t lda) [inline]
```

**12.217.3.10 checkB()**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
bool checkB (
    const Field & F,
    const FFLAS::FFLAS_TRANSPOSE tb,
    const size_t M,
    const size_t N,
    typename Field::ConstElement_ptr B,
    const size_t ldb) [inline]
```

**12.217.3.11 checkOut() [1/2]**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
bool checkOut (
    const Field & F,
    const size_t M,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t lda) [inline]
```

**12.217.3.12 checkOut() [2/2]**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
bool checkOut (
```

```

    const Field & F,
    FFLAS_UPLO uplo,
    const size_t M,
    const size_t N,
    typename Field::ConstElement_ptr A,
    const size_t lda) [inline]

```

## 12.217.4 Friends And Related Symbol Documentation

### 12.217.4.1 operator<<

```

template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
std::ostream & operator<< (
    std::ostream & out,
    const Self_t & M) [friend]

```

## 12.217.5 Field Documentation

### 12.217.5.1 recLevel

```

template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
int recLevel

```

### 12.217.5.2 FieldMin

```

template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
DFElt FieldMin

```

### 12.217.5.3 FieldMax

```

template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
DFElt FieldMax

```

### 12.217.5.4 Amin

```

template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
DFElt Amin

```

### 12.217.5.5 Amax

```

template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
DFElt Amax

```

### 12.217.5.6 Bmin

```

template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
DFElt Bmin

```

### 12.217.5.7 Bmax

```

template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
DFElt Bmax

```

### 12.217.5.8 Cmin

```

template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
DFElt Cmin

```

#### 12.217.5.9 Cmax

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
DFelt Cmax
```

#### 12.217.5.10 Outmin

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
DFelt Outmin
```

#### 12.217.5.11 Outmax

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
DFelt Outmax
```

#### 12.217.5.12 MaxStorableValue

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
DFelt MaxStorableValue
```

#### 12.217.5.13 delayedField

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
const DelayedField_t delayedField
```

#### 12.217.5.14 parseq

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
ParSeqTrait parseq
```

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

## 12.218 MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait > Struct Template Reference

### Public Types

- typedef [MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >](#) [Self\\_t](#)

### Public Member Functions

- [MMHelper](#) ()
- [MMHelper](#) (Givaro::Integer Amax, Givaro::Integer Bmax)
- [MMHelper](#) (const [FFPACK::RNSInteger< E >](#) &F, size\_t m, size\_t n, size\_t k, ParSeqTrait PS=ParSeqTrait())
- [MMHelper](#) (const [FFPACK::RNSInteger< E >](#) &F, int wino, ParSeqTrait PS=ParSeqTrait())
- template<class F2 , class A2 , class M2 , class PS2 >  
[MMHelper](#) ([MMHelper< F2, A2, M2, PS2 >](#) H2)
- void [setNorm](#) (Givaro::Integer p)

### Data Fields

- Givaro::Integer [normA](#)
- Givaro::Integer [normB](#)
- int [recLevel](#)
- ParSeqTrait [parseq](#)

## Friends

- `std::ostream & operator<< (std::ostream &out, const Self_t &M)`

## 12.218.1 Member Typedef Documentation

### 12.218.1.1 Self\_t

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
MMHelper<FFPACK::RNSInteger<E>, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait> Self_t
```

## 12.218.2 Constructor & Destructor Documentation

### 12.218.2.1 MMHelper() [1/5]

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
MMHelper () [inline]
```

### 12.218.2.2 MMHelper() [2/5]

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
MMHelper (
    Givaro::Integer Amax,
    Givaro::Integer Bmax) [inline]
```

### 12.218.2.3 MMHelper() [3/5]

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
MMHelper (
    const FFPACK::RNSInteger< E > & F,
    size_t m,
    size_t n,
    size_t k,
    ParSeqTrait PS = ParSeqTrait()) [inline]
```

### 12.218.2.4 MMHelper() [4/5]

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
MMHelper (
    const FFPACK::RNSInteger< E > & F,
    int wino,
    ParSeqTrait PS = ParSeqTrait()) [inline]
```

### 12.218.2.5 MMHelper() [5/5]

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
template<class F2 , class A2 , class M2 , class PS2 >
MMHelper (
    MMHelper< F2, A2, M2, PS2 > H2) [inline]
```

## 12.218.3 Member Function Documentation

### 12.218.3.1 setNorm()

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
void setNorm (
    Givaro::Integer p) [inline]
```

## 12.218.4 Friends And Related Symbol Documentation

### 12.218.4.1 operator<<

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
std::ostream & operator<< (
    std::ostream & out,
    const Self_t & M) [friend]
```

## 12.218.5 Field Documentation

### 12.218.5.1 normA

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
Givaro::Integer normA
```

### 12.218.5.2 normB

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
Givaro::Integer normB
```

### 12.218.5.3 recLevel

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
int recLevel
```

### 12.218.5.4 parseq

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
ParSeqTrait parseq
```

The documentation for this struct was generated from the following file:

- [fgemm\\_classical\\_mp.inl](#)

## 12.219 MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait > Struct Template Reference

### Public Types

- typedef [MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait > Self\\_t](#)

### Public Member Functions

- [MMHelper](#) ()
- [MMHelper](#) (Givaro::Integer Amax, Givaro::Integer Bmax)
- [MMHelper](#) (const [FFPACK::RNSIntegerMod< E >](#) &F, size\_t m, size\_t n, size\_t k, ParSeqTrait PS=ParSeqTrait())
- [MMHelper](#) (const [FFPACK::RNSIntegerMod< E >](#) &F, int wino, ParSeqTrait PS=ParSeqTrait())
- template<class F2 , typename AlgoT2 , typename FT2 , typename PS2 > [MMHelper](#) ([MMHelper](#)< F2, AlgoT2, FT2, PS2 > &WH)
- void [setNorm](#) (Givaro::Integer p)

### Data Fields

- Givaro::Integer [normA](#)
- Givaro::Integer [normB](#)
- int [recLevel](#)
- ParSeqTrait [parseq](#)

## Friends

- `std::ostream & operator<< (std::ostream &out, const Self_t &M)`

## 12.219.1 Member Typedef Documentation

### 12.219.1.1 Self\_t

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
MMHelper<FFPACK::RNSIntegerMod<E>, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait> Self_t
```

## 12.219.2 Constructor & Destructor Documentation

### 12.219.2.1 MMHelper() [1/5]

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
MMHelper () [inline]
```

### 12.219.2.2 MMHelper() [2/5]

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
MMHelper (
    Givaro::Integer Amax,
    Givaro::Integer Bmax) [inline]
```

### 12.219.2.3 MMHelper() [3/5]

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
MMHelper (
    const FFPACK::RNSIntegerMod< E > & F,
    size_t m,
    size_t n,
    size_t k,
    ParSeqTrait PS = ParSeqTrait()) [inline]
```

### 12.219.2.4 MMHelper() [4/5]

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
MMHelper (
    const FFPACK::RNSIntegerMod< E > & F,
    int wino,
    ParSeqTrait PS = ParSeqTrait()) [inline]
```

### 12.219.2.5 MMHelper() [5/5]

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
template<class F2 , typename AlgoT2 , typename FT2 , typename PS2 >
MMHelper (
    MMHelper< F2, AlgoT2, FT2, PS2 > & WH) [inline]
```

## 12.219.3 Member Function Documentation

### 12.219.3.1 setNorm()

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
void setNorm (
    Givaro::Integer p) [inline]
```



## 12.219.4 Friends And Related Symbol Documentation

### 12.219.4.1 operator<<

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
std::ostream & operator<< (
    std::ostream & out,
    const Self_t & M) [friend]
```

## 12.219.5 Field Documentation

### 12.219.5.1 normA

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
Givaro::Integer normA
```

### 12.219.5.2 normB

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
Givaro::Integer normB
```

### 12.219.5.3 recLevel

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
int recLevel
```

### 12.219.5.4 parseq

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
ParSeqTrait parseq
```

The documentation for this struct was generated from the following file:

- [fgemm\\_classical\\_mp.inl](#)

## 12.220 MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< Dest >, ParSeqTrait > Struct Template Reference

### Public Types

- typedef [MMHelper](#)< [Field](#), [AlgoTrait](#), [ModeCategories::ConvertTo](#)< [Dest](#) >, [ParSeqTrait](#) > [Self\\_t](#)

### Public Member Functions

- [MMHelper](#) ()
- [MMHelper](#) (const [Field](#) &F, size\_t m, size\_t k, size\_t n, [ParSeqTrait](#) \_PS)
- [MMHelper](#) (const [Field](#) &F, int w, [ParSeqTrait](#) \_PS=[ParSeqTrait](#)())
- template<class F2 , typename AlgoT2 , typename FT2 , typename PS2 > [MMHelper](#) ([MMHelper](#)< F2, AlgoT2, FT2, PS2 > &WH)

### Data Fields

- int [recLevel](#)
- [ParSeqTrait](#) [parseq](#)

### Friends

- std::ostream & [operator<<](#) (std::ostream &out, const [Self\\_t](#) &M)

## 12.220.1 Member Typedef Documentation

### 12.220.1.1 Self\_t

```
template<class Field , typename AlgoTrait , typename Dest , typename ParSeqTrait >
MMHelper<Field,AlgoTrait, ModeCategories::ConvertTo<Dest>,ParSeqTrait> Self_t
```

## 12.220.2 Constructor & Destructor Documentation

### 12.220.2.1 MMHelper() [1/4]

```
template<class Field , typename AlgoTrait , typename Dest , typename ParSeqTrait >
MMHelper () [inline]
```

### 12.220.2.2 MMHelper() [2/4]

```
template<class Field , typename AlgoTrait , typename Dest , typename ParSeqTrait >
MMHelper (
    const Field & F,
    size_t m,
    size_t k,
    size_t n,
    ParSeqTrait _PS) [inline]
```

### 12.220.2.3 MMHelper() [3/4]

```
template<class Field , typename AlgoTrait , typename Dest , typename ParSeqTrait >
MMHelper (
    const Field & F,
    int w,
    ParSeqTrait _PS = ParSeqTrait()) [inline]
```

### 12.220.2.4 MMHelper() [4/4]

```
template<class Field , typename AlgoTrait , typename Dest , typename ParSeqTrait >
template<class F2 , typename AlgoT2 , typename FT2 , typename PS2 >
MMHelper (
    MMHelper< F2, AlgoT2, FT2, PS2 > & WH) [inline]
```

## 12.220.3 Friends And Related Symbol Documentation

### 12.220.3.1 operator<<

```
template<class Field , typename AlgoTrait , typename Dest , typename ParSeqTrait >
std::ostream & operator<< (
    std::ostream & out,
    const Self_t & M) [friend]
```

## 12.220.4 Field Documentation

### 12.220.4.1 recLevel

```
template<class Field , typename AlgoTrait , typename Dest , typename ParSeqTrait >
int recLevel
```

### 12.220.4.2 parseq

```
template<class Field , typename AlgoTrait , typename Dest , typename ParSeqTrait >
ParSeqTrait parseq
```

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

## 12.221 MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeqTrait > Struct Template Reference

### Public Types

- typedef [MMHelper](#)< [Field](#), [AlgoTrait](#), [ModeCategories::ConvertTo](#)< [ElementCategories::RNSElementTag](#) >, [ParSeqTrait](#) > [Self\\_t](#)

### Public Member Functions

- [MMHelper](#) ()
- template<class F2 , class A2 , class M2 , class PS2 >  
[MMHelper](#) ([MMHelper](#)< F2, A2, M2, PS2 > H2)
- [MMHelper](#) (Givaro::Integer Amax, Givaro::Integer Bmax)
- [MMHelper](#) (const [Field](#) &F, size\_t m, size\_t n, size\_t k, [ParSeqTrait](#) PS=[ParSeqTrait](#)())
- [MMHelper](#) (const [Field](#) &F, int wino, [ParSeqTrait](#) PS=[ParSeqTrait](#)())
- void [setNorm](#) (Givaro::Integer p)

### Data Fields

- Givaro::Integer [normA](#)
- Givaro::Integer [normB](#)
- int [recLevel](#)
- [ParSeqTrait](#) [parseq](#)

### Friends

- std::ostream & [operator<<](#) (std::ostream &out, const [Self\\_t](#) &M)

## 12.221.1 Member Typedef Documentation

### 12.221.1.1 Self\_t

```
template<typename Field , typename AlgoTrait , typename ParSeqTrait >
MMHelper<Field, AlgoTrait,ModeCategories::ConvertTo<ElementCategories::RNSElementTag>, ParSeqTrait> Self\_t
```

## 12.221.2 Constructor & Destructor Documentation

### 12.221.2.1 MMHelper() [1/5]

```
template<typename Field , typename AlgoTrait , typename ParSeqTrait >
MMHelper () [inline]
```

### 12.221.2.2 MMHelper() [2/5]

```
template<typename Field , typename AlgoTrait , typename ParSeqTrait >
template<class F2 , class A2 , class M2 , class PS2 >
MMHelper (
    MMHelper< F2, A2, M2, PS2 > H2) [inline]
```

### 12.221.2.3 MMHelper() [3/5]

```
template<typename Field , typename AlgoTrait , typename ParSeqTrait >
MMHelper (
    Givaro::Integer Amax,
    Givaro::Integer Bmax) [inline]
```

**12.221.2.4 MMHelper() [4/5]**

```
template<typename Field , typename AlgoTrait , typename ParSeqTrait >
MMHelper (
    const Field & F,
    size_t m,
    size_t n,
    size_t k,
    ParSeqTrait PS = ParSeqTrait()) [inline]
```

**12.221.2.5 MMHelper() [5/5]**

```
template<typename Field , typename AlgoTrait , typename ParSeqTrait >
MMHelper (
    const Field & F,
    int wino,
    ParSeqTrait PS = ParSeqTrait()) [inline]
```

**12.221.3 Member Function Documentation****12.221.3.1 setNorm()**

```
template<typename Field , typename AlgoTrait , typename ParSeqTrait >
void setNorm (
    Givaro::Integer p) [inline]
```

**12.221.4 Friends And Related Symbol Documentation****12.221.4.1 operator<<**

```
template<typename Field , typename AlgoTrait , typename ParSeqTrait >
std::ostream & operator<< (
    std::ostream & out,
    const Self_t & M) [friend]
```

**12.221.5 Field Documentation****12.221.5.1 normA**

```
template<typename Field , typename AlgoTrait , typename ParSeqTrait >
Givaro::Integer normA
```

**12.221.5.2 normB**

```
template<typename Field , typename AlgoTrait , typename ParSeqTrait >
Givaro::Integer normB
```

**12.221.5.3 recLevel**

```
template<typename Field , typename AlgoTrait , typename ParSeqTrait >
int recLevel
```

**12.221.5.4 parseq**

```
template<typename Field , typename AlgoTrait , typename ParSeqTrait >
ParSeqTrait parseq
```

The documentation for this struct was generated from the following file:

- [fgemm\\_classical\\_mp.inl](#)

## 12.222 MMHelper< Field, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait > Struct Template Reference

FGEMM Helper for Default and ConvertTo modes of operation.

### Public Types

- typedef [MMHelper](#)< [Field](#), [AlgoTrait](#), [ModeCategories::DefaultTag](#), [ParSeqTrait](#) > [Self\\_t](#)

### Public Member Functions

- [MMHelper](#) ()
- [MMHelper](#) (const [Field](#) &F, size\_t m, size\_t k, size\_t n, [ParSeqTrait](#) \_PS)
- [MMHelper](#) (const [Field](#) &F, int w, [ParSeqTrait](#) \_PS=[ParSeqTrait](#)())
- template<class F2, typename AlgoT2, typename FT2, typename PS2 >  
[MMHelper](#) ([MMHelper](#)< F2, AlgoT2, FT2, PS2 > &WH)

### Data Fields

- int [recLevel](#)
- [ParSeqTrait](#) [parseq](#)

### Friends

- std::ostream & [operator](#)<< (std::ostream &out, const [Self\\_t](#) &M)

### 12.222.1 Detailed Description

```
template<class Field, typename AlgoTrait, typename ParSeqTrait>
struct FFLAS::MMHelper< Field, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >
```

FGEMM Helper for Default and ConvertTo modes of operation.

### 12.222.2 Member Typedef Documentation

#### 12.222.2.1 Self\_t

```
template<class Field , typename AlgoTrait , typename ParSeqTrait >
MMHelper<Field,AlgoTrait, ModeCategories::DefaultTag,ParSeqTrait> Self\_t
```

### 12.222.3 Constructor & Destructor Documentation

#### 12.222.3.1 MMHelper() [1/4]

```
template<class Field , typename AlgoTrait , typename ParSeqTrait >
MMHelper () [inline]
```

#### 12.222.3.2 MMHelper() [2/4]

```
template<class Field , typename AlgoTrait , typename ParSeqTrait >
MMHelper (
    const Field & F,
    size_t m,
    size_t k,
    size_t n,
    ParSeqTrait _PS) [inline]
```

**12.222.3.3 MMHelper() [3/4]**

```
template<class Field , typename AlgoTrait , typename ParSeqTrait >
MMHelper (
    const Field & F,
    int w,
    ParSeqTrait _PS = ParSeqTrait()) [inline]
```

**12.222.3.4 MMHelper() [4/4]**

```
template<class Field , typename AlgoTrait , typename ParSeqTrait >
template<class F2 , typename AlgoT2 , typename FT2 , typename PS2 >
MMHelper (
    MMHelper< F2, AlgoT2, FT2, PS2 > & WH) [inline]
```

**12.222.4 Friends And Related Symbol Documentation****12.222.4.1 operator<<**

```
template<class Field , typename AlgoTrait , typename ParSeqTrait >
std::ostream & operator<< (
    std::ostream & out,
    const Self_t & M) [friend]
```

**12.222.5 Field Documentation****12.222.5.1 recLevel**

```
template<class Field , typename AlgoTrait , typename ParSeqTrait >
int recLevel
```

**12.222.5.2 parseq**

```
template<class Field , typename AlgoTrait , typename ParSeqTrait >
ParSeqTrait parseq
```

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

**12.223 ModeTraits< Field > Struct Template Reference**

[ModeTraits.](#)

```
#include <field-traits.h>
```

**Public Types**

- typedef [ModeCategories::DefaultTag](#) value

**12.223.1 Detailed Description**

```
template<class Field>
struct FFLAS::ModeTraits< Field >
```

[ModeTraits.](#)

**12.223.2 Member Typedef Documentation****12.223.2.1 value**

```
template<class Field >
ModeCategories::DefaultTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.224 ModeTraits< Givaro::Modular< Element, Compute > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ModeCategories::DelayedTag](#) value

### 12.224.1 Member Typedef Documentation

#### 12.224.1.1 value

```
template<typename Element , typename Compute >
```

```
ModeCategories::DelayedTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.225 ModeTraits< Givaro::Modular< Givaro::Integer, Compute > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ModeCategories::ConvertTo< ElementCategories::RNSElementTag >](#) value

### 12.225.1 Member Typedef Documentation

#### 12.225.1.1 value

```
template<typename Compute >
```

```
ModeCategories::ConvertTo<ElementCategories::RNSElementTag> value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.226 ModeTraits< Givaro::Modular< int16\_t, Compute > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ModeCategories::ConvertTo< ElementCategories::MachineFloatTag >](#) value

### 12.226.1 Member Typedef Documentation

#### 12.226.1.1 value

```
template<typename Compute >
```

```
ModeCategories::ConvertTo<ElementCategories::MachineFloatTag> value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.227 ModeTraits< Givaro::Modular< int32\_t, Compute > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ModeCategories::ConvertTo< ElementCategories::MachineFloatTag > value](#)

### 12.227.1 Member Typedef Documentation

#### 12.227.1.1 value

```
template<typename Compute >
```

```
ModeCategories::ConvertTo<ElementCategories::MachineFloatTag> value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.228 ModeTraits< Givaro::Modular< int64\_t, uint64\_t > > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ModeCategories::DefaultTag value](#)

### 12.228.1 Member Typedef Documentation

#### 12.228.1.1 value

```
ModeCategories::DefaultTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.229 ModeTraits< Givaro::Modular< int8\_t, Compute > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ModeCategories::ConvertTo< ElementCategories::MachineFloatTag > value](#)

### 12.229.1 Member Typedef Documentation

#### 12.229.1.1 value

```
template<typename Compute >
```

```
ModeCategories::ConvertTo<ElementCategories::MachineFloatTag> value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)



## 12.230 ModeTraits< Givaro::Modular< RecInt::ruint< K >, Compute > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ModeCategories::ConvertTo< ElementCategories::RNSElementTag > value](#)

### 12.230.1 Member Typedef Documentation

#### 12.230.1.1 value

```
template<typename Compute , size_t K>
```

```
ModeCategories::ConvertTo<ElementCategories::RNSElementTag> value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.231 ModeTraits< Givaro::Modular< uint16\_t, Compute > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ModeCategories::ConvertTo< ElementCategories::MachineFloatTag > value](#)

### 12.231.1 Member Typedef Documentation

#### 12.231.1.1 value

```
template<typename Compute >
```

```
ModeCategories::ConvertTo<ElementCategories::MachineFloatTag> value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.232 ModeTraits< Givaro::Modular< uint32\_t, Compute > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ModeCategories::ConvertTo< ElementCategories::MachineFloatTag > value](#)

### 12.232.1 Member Typedef Documentation

#### 12.232.1.1 value

```
template<typename Compute >
```

```
ModeCategories::ConvertTo<ElementCategories::MachineFloatTag> value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.233 ModeTraits< Givaro::Modular< uint8\_t, Compute > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ModeCategories::ConvertTo< ElementCategories::MachineFloatTag > value](#)

### 12.233.1 Member Typedef Documentation

#### 12.233.1.1 value

```
template<typename Compute >
```

```
ModeCategories::ConvertTo<ElementCategories::MachineFloatTag> value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.234 ModeTraits< Givaro::ModularBalanced< Element > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ModeCategories::DelayedTag value](#)

### 12.234.1 Member Typedef Documentation

#### 12.234.1.1 value

```
template<typename Element >
```

```
ModeCategories::DelayedTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.235 ModeTraits< Givaro::ModularBalanced< Givaro::Integer > > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ModeCategories::ConvertTo< ElementCategories::RNSElementTag > value](#)

### 12.235.1 Member Typedef Documentation

#### 12.235.1.1 value

```
ModeCategories::ConvertTo<ElementCategories::RNSElementTag> value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.236 ModeTraits< Givaro::ModularBalanced< int16\_t > > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ModeCategories::ConvertTo< ElementCategories::MachineFloatTag > value](#)

### 12.236.1 Member Typedef Documentation

#### 12.236.1.1 value

[ModeCategories::ConvertTo<ElementCategories::MachineFloatTag> value](#)

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.237 ModeTraits< Givaro::ModularBalanced< int32\_t > > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ModeCategories::ConvertTo< ElementCategories::MachineFloatTag > value](#)

### 12.237.1 Member Typedef Documentation

#### 12.237.1.1 value

[ModeCategories::ConvertTo<ElementCategories::MachineFloatTag> value](#)

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.238 ModeTraits< Givaro::ModularBalanced< int8\_t > > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- typedef [ModeCategories::ConvertTo< ElementCategories::MachineFloatTag > value](#)

### 12.238.1 Member Typedef Documentation

#### 12.238.1.1 value

[ModeCategories::ConvertTo<ElementCategories::MachineFloatTag> value](#)

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.239 ModeTraits< Givaro::Montgomery< T > > Struct Template Reference

```
#include <field-traits.h>
```

**Public Types**

- typedef [ModeCategories::DefaultBoundedTag](#) value

**12.239.1 Member Typedef Documentation****12.239.1.1 value**

```
template<class T >
```

```
ModeCategories::DefaultBoundedTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

**12.240 ModeTraits< Givaro::ZRing< double > > Struct Reference**

```
#include <field-traits.h>
```

**Public Types**

- typedef [ModeCategories::DefaultBoundedTag](#) value

**12.240.1 Member Typedef Documentation****12.240.1.1 value**

```
ModeCategories::DefaultBoundedTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

**12.241 ModeTraits< Givaro::ZRing< float > > Struct Reference**

```
#include <field-traits.h>
```

**Public Types**

- typedef [ModeCategories::DefaultBoundedTag](#) value

**12.241.1 Member Typedef Documentation****12.241.1.1 value**

```
ModeCategories::DefaultBoundedTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

**12.242 ModeTraits< Givaro::ZRing< Givaro::Integer > > Struct Reference**

```
#include <field-traits.h>
```

**Public Types**

- typedef [ModeCategories::ConvertTo< ElementCategories::RNSElementTag >](#) value

### 12.242.1 Member Typedef Documentation

#### 12.242.1.1 value

`ModeCategories::ConvertTo<ElementCategories::RNSElementTag> value`

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.243 ModularBalanced< T > Class Template Reference

The documentation for this class was generated from the following file:

- [field-traits.h](#)

### 12.244 ModularTag Struct Reference

This is a modular field like e.g. `Modular<T>` or `ModularBalanced<T>`  
`#include <field-traits.h>`

#### 12.244.1 Detailed Description

This is a modular field like e.g. `Modular<T>` or `ModularBalanced<T>`  
 The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.245 Montgomery< T > Class Template Reference

The documentation for this class was generated from the following file:

- [field-traits.h](#)

## 12.246 need\_field\_characteristic< Field > Struct Template Reference

#### Static Public Attributes

- static constexpr bool [value](#) = false

### 12.246.1 Field Documentation

#### 12.246.1.1 value

```
template<typename Field >
bool value = false [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [benchmark-fgemv.C](#)

## 12.247 need\_field\_characteristic< Givaro::Modular< Field > > Struct Template Reference

#### Static Public Attributes

- static constexpr bool [value](#) = true

## 12.247.1 Field Documentation

### 12.247.1.1 value

```
template<typename Field >
bool value = true [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [benchmark-fgemv.C](#)

## 12.248 need\_field\_characteristic< Givaro::ModularBalanced< Field > > Struct Template Reference

### Static Public Attributes

- static constexpr bool [value](#) = true

## 12.248.1 Field Documentation

### 12.248.1.1 value

```
template<typename Field >
bool value = true [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [benchmark-fgemv.C](#)

## 12.249 NoSimd< T > Struct Template Reference

```
#include <fflas_simd.h>
```

### Public Types

- using [vect\\_t](#) = T\*
- using [scalar\\_t](#) = T
- using [aligned\\_allocator](#) = AlignedAllocator<[scalar\\_t](#), Alignment([alignment](#))>
- using [aligned\\_vector](#) = std::vector<[scalar\\_t](#), [aligned\\_allocator](#)>
- template<class [Field](#) >
 using [is\\_same\\_element](#) = std::is\_same<typename [Field::Element](#), T>

### Static Public Member Functions

- static const std::string [type\\_string](#) ()
- template<class TT >
 static constexpr bool [valid](#) (TT p)
- template<class TT >
 static constexpr bool [compliant](#) (TT n)

### Static Public Attributes

- static const constexpr size\_t [vect\\_size](#) = 1
- static const constexpr size\_t [alignment](#) = static\_cast<size\_t>(Alignment::Normal)

## 12.249.1 Member Typedef Documentation

### 12.249.1.1 vect\_t

```
template<typename T >
using vect\_t = T*
```

### 12.249.1.2 scalar\_t

```
template<typename T >
using scalar_t = T
```

### 12.249.1.3 aligned\_allocator

```
template<typename T >
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

### 12.249.1.4 aligned\_vector

```
template<typename T >
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

### 12.249.1.5 is\_same\_element

```
template<typename T >
template<class Field >
using is_same_element = std::is_same<typename Field::Element, T>
```

## 12.249.2 Member Function Documentation

### 12.249.2.1 type\_string()

```
template<typename T >
static const std::string type_string () [inline], [static]
```

### 12.249.2.2 valid()

```
template<typename T >
template<class TT >
static constexpr bool valid (
    TT p) [inline], [static], [constexpr]
```

### 12.249.2.3 compliant()

```
template<typename T >
template<class TT >
static constexpr bool compliant (
    TT n) [inline], [static], [constexpr]
```

## 12.249.3 Field Documentation

### 12.249.3.1 vect\_size

```
template<typename T >
const constexpr size_t vect_size = 1 [static], [constexpr]
```

### 12.249.3.2 alignment

```
template<typename T >
const constexpr size_t alignment = static_cast<size_t>(Alignment::Normal) [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [fflas\\_simd.h](#)

## 12.250 Parallel< C, P > Struct Template Reference

### Public Types

- typedef C [Cut](#)
- typedef P [Param](#)

### Public Member Functions

- [Parallel](#) (size\_t n=NUM\_THREADS)
- size\_t [numthreads](#) () const
- size\_t & [set\\_numthreads](#) (size\_t n)

### Friends

- std::ostream & [operator<<](#) (std::ostream &out, const [Parallel](#) &p)

## 12.250.1 Member Typedef Documentation

### 12.250.1.1 Cut

```
template<typename C = CuttingStrategy::Block, typename P = StrategyParameter::Threads>
C Cut
```

### 12.250.1.2 Param

```
template<typename C = CuttingStrategy::Block, typename P = StrategyParameter::Threads>
P Param
```

## 12.250.2 Constructor & Destructor Documentation

### 12.250.2.1 Parallel()

```
template<typename C = CuttingStrategy::Block, typename P = StrategyParameter::Threads>
Parallel (
    size_t n = NUM_THREADS) [inline]
```

## 12.250.3 Member Function Documentation

### 12.250.3.1 numthreads()

```
template<typename C = CuttingStrategy::Block, typename P = StrategyParameter::Threads>
size_t numthreads () const [inline]
```

### 12.250.3.2 set\_numthreads()

```
template<typename C = CuttingStrategy::Block, typename P = StrategyParameter::Threads>
size_t & set_numthreads (
    size_t n) [inline]
```

## 12.250.4 Friends And Related Symbol Documentation

### 12.250.4.1 operator<<

```
template<typename C = CuttingStrategy::Block, typename P = StrategyParameter::Threads>
std::ostream & operator<< (
    std::ostream & out,
    const Parallel< C, P > & p) [friend]
```

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)



## 12.251 RNSInteger< RNS >::RandIter Class Reference

#include <rns-integer.h>

Inheritance diagram for RNSInteger< RNS >::RandIter:



### Public Member Functions

- [RandIter](#) (const [RNSInteger< RNS >](#) &F, uint64\_t seed=0)
- [RNS::Element](#) & [random](#) (typename [RNS::Element](#) &elt) const  
*RNS ring Element random assignment.*
- [RNS::Element](#) [random](#) () const
- [RNS::Element](#) & [operator\(\)](#) (typename [RNS::Element](#) &elt) const
- [RNS::Element](#) [operator\(\)](#) () const
- const [RNS](#) & [ring](#) () const

### 12.251.1 Constructor & Destructor Documentation

#### 12.251.1.1 RandIter()

```

template<typename RNS >
RandIter (
    const RNSInteger< RNS > & F,
    uint64_t seed = 0) [inline]
  
```

### 12.251.2 Member Function Documentation

#### 12.251.2.1 random() [1/2]

```

template<typename RNS >
RNS::Element & random (
    typename RNS::Element & elt) const [inline], [inherited]
  
```

RNS ring Element random assignment.

Element is supposed to be initialized

Returns

random ring Element

#### 12.251.2.2 random() [2/2]

```

template<typename RNS >
RNS::Element random () const [inline], [inherited]
  
```

#### 12.251.2.3 operator>() [1/2]

```

template<typename RNS >
RNS::Element & operator() (
    typename RNS::Element & elt) const [inline], [inherited]
  
```

#### 12.251.2.4 operator>() [2/2]

```

template<typename RNS >
RNS::Element operator() () const [inline], [inherited]
  
```

### 12.251.2.5 ring()

```
template<typename RNS >
```

```
const RNS & ring () const [inline], [inherited]
```

The documentation for this class was generated from the following file:

- [rns-integer.h](#)

## 12.252 RNSIntegerMod< RNS >::RandIter Class Reference

```
#include <rns-integer-mod.h>
```

Inheritance diagram for RNSIntegerMod< RNS >::RandIter:



### Public Member Functions

- [RandIter](#) (const [RNSIntegerMod< RNS >](#) &F, uint64\_t seed=0)
- [RNS::Element](#) & [random](#) (typename [RNS::Element](#) &elt) const
- [RNS::Element](#) [random](#) () const
- [RNS::Element](#) & [operator\(\)](#) (typename [RNS::Element](#) &elt) const
- [RNS::Element](#) [operator\(\)](#) () const
- const [RNS](#) & [ring](#) () const

### 12.252.1 Constructor & Destructor Documentation

#### 12.252.1.1 RandIter()

```
template<typename RNS >
```

```
RandIter (
    const RNSIntegerMod< RNS > & F,
    uint64_t seed = 0) [inline]
```

### 12.252.2 Member Function Documentation

#### 12.252.2.1 random() [1/2]

```
template<typename RNS >
```

```
RNS::Element & random (
    typename RNS::Element & elt) const [inline]
```

#### 12.252.2.2 random() [2/2]

```
template<typename RNS >
```

```
RNS::Element random () const [inline], [inherited]
```

#### 12.252.2.3 operator>() [1/2]

```
template<typename RNS >
```

```
RNS::Element & operator() (
    typename RNS::Element & elt) const [inline], [inherited]
```

#### 12.252.2.4 operator>() [2/2]

```
template<typename RNS >
```

```
RNS::Element operator() () const [inline], [inherited]
```

### 12.252.2.5 ring()

```
template<typename RNS >
```

```
const RNS & ring () const [inline], [inherited]
```

The documentation for this class was generated from the following file:

- [rns-integer-mod.h](#)

## 12.253 readMyMachineType< Field, T > Struct Template Reference

```
#include <read_sparse.h>
```

### Public Types

- typedef [Field::Element](#) [Element](#)
- typedef [Field::Element\\_ptr](#) [Element\\_ptr](#)

### Public Member Functions

- void [operator\(\)](#) (const [Field](#) &F, [Element](#) &modulo, [Element\\_ptr](#) val, std::ifstream &file, const uint64\_t dims, const [mask\\_t](#) data\_type, const [mask\\_t](#) field\_desc)

### 12.253.1 Member Typedef Documentation

#### 12.253.1.1 Element

```
template<class Field , class T >
```

```
Field::Element Element
```

#### 12.253.1.2 Element\_ptr

```
template<class Field , class T >
```

```
Field::Element_ptr Element_ptr
```

### 12.253.2 Member Function Documentation

#### 12.253.2.1 operator>()()

```
template<class Field , typename T >
```

```
void operator() (
    const Field & F,
    Element & modulo,
    Element_ptr val,
    std::ifstream & file,
    const uint64_t dims,
    const mask_t data_type,
    const mask_t field_desc)
```

The documentation for this struct was generated from the following file:

- [read\\_sparse.h](#)

## 12.254 readMyMachineType< Field, mpz\_t > Struct Template Reference

```
#include <read_sparse.h>
```

### Public Types

- typedef [Field::Element](#) [Element](#)
- typedef [Field::Element\\_ptr](#) [Element\\_ptr](#)

## Public Member Functions

- void [operator\(\)](#) (const [Field](#) &F, [Element](#) &modulo, [Element\\_ptr](#) val, std::ifstream &file, const uint64\_t dims, const [mask\\_t](#) data\_type, const [mask\\_t](#) field\_desc)

## 12.254.1 Member Typedef Documentation

### 12.254.1.1 Element

```
template<class Field >
Field::Element Element
```

### 12.254.1.2 Element\_ptr

```
template<class Field >
Field::Element\_ptr Element\_ptr
```

## 12.254.2 Member Function Documentation

### 12.254.2.1 operator()()

```
template<class Field >
void operator\(\) (
    const Field & F,
    typename Field::Element & modulo,
    typename Field::Element\_ptr val,
    std::ifstream & file,
    const uint64_t dims,
    const mask\_t data_type,
    const mask\_t field_desc)
```

The documentation for this struct was generated from the following file:

- [read\\_sparse.h](#)

## 12.255 Recursive Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.256 Recursive Struct Reference

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

## 12.257 rint< K > Class Template Reference

The documentation for this class was generated from the following file:

- [field-traits.h](#)

## 12.258 rns\_double Struct Reference

```
#include <rns-double.h>
```

## Public Types

- typedef Givaro::Integer [integer](#)
- typedef Givaro::Modular< double > [ModField](#)
- typedef double [BasisElement](#)
- typedef [rns\\_double\\_elt](#) [Element](#)
- typedef [rns\\_double\\_elt\\_ptr](#) [Element\\_ptr](#)
- typedef [rns\\_double\\_elt\\_cstptr](#) [ConstElement\\_ptr](#)

## Public Member Functions

- [rns\\_double](#) (const [integer](#) &bound, size\_t pbits, bool rnsmod=false, long seed=time(NULL))
- [rns\\_double](#) (size\_t pbits, size\_t size, long seed=time(NULL))
- template<typename Vect >  
  [rns\\_double](#) (const Vect &basis, bool rnsmod=false, long seed=time(NULL))
- [rns\\_double](#) (const [RNSIntegerMod](#)< [rns\\_double](#) > &basis, bool rnsmod=false, long seed=time(NULL))
- void [precompute\\_cst](#) (size\_t K=0)
- template<typename T >  
  void [init](#) (size\_t m, size\_t n, double \*Arns, size\_t rda, const T \*A, size\_t lda, const [integer](#) &maxA, bool RNS\_MAJOR=false) const
- void [init](#) (size\_t m, size\_t n, double \*Arns, size\_t rda, const [integer](#) \*A, size\_t lda, size\_t k, bool RNS\_MAJOR=false) const
- void [init\\_transpose](#) (size\_t m, size\_t n, double \*Arns, size\_t rda, const [integer](#) \*A, size\_t lda, size\_t k, bool RNS\_MAJOR=false) const
- void [convert](#) (size\_t m, size\_t n, [integer](#) gamma, [integer](#) \*A, size\_t lda, const double \*Arns, size\_t rda, bool RNS\_MAJOR=false) const
- void [convert\\_transpose](#) (size\_t m, size\_t n, [integer](#) gamma, [integer](#) \*A, size\_t lda, const double \*Arns, size\_t rda, bool RNS\_MAJOR=false) const
- void [reduce](#) (size\_t n, double \*Arns, size\_t rda, bool RNS\_MAJOR=false) const
- template<size\_t K>  
  void [init](#) (size\_t m, size\_t n, double \*Arns, size\_t rda, const [Reclnt::ruint](#)< K > \*A, size\_t lda, size\_t k, bool RNS\_MAJOR=false) const
- template<size\_t K>  
  void [convert](#) (size\_t m, size\_t n, [integer](#) gamma, [Reclnt::ruint](#)< K > \*A, size\_t lda, const double \*Arns, size\_t rda, [integer](#) p=0, bool RNS\_MAJOR=false) const

## Data Fields

- std::vector< double, AlignedAllocator< double, Alignment::CACHE\_LINE > > [\\_basis](#)
- std::vector< double, AlignedAllocator< double, Alignment::CACHE\_LINE > > [\\_basisMax](#)
- std::vector< double, AlignedAllocator< double, Alignment::CACHE\_LINE > > [\\_negbasis](#)
- std::vector< double, AlignedAllocator< double, Alignment::CACHE\_LINE > > [\\_invbasis](#)
- std::vector< [ModField](#) > [\\_field\\_rns](#)
- [integer](#) [\\_M](#)
- std::vector< [integer](#) > [\\_Mi](#)
- std::vector< double > [\\_MMi](#)
- std::vector< double > [\\_crt\\_in](#)
- std::vector< double > [\\_crt\\_out](#)
- size\_t [\\_size](#)
- size\_t [\\_pbits](#)
- size\_t [\\_ldm](#)
- [integer](#) [\\_mi\\_sum](#)

## 12.258.1 Member Typedef Documentation

### 12.258.1.1 integer

Givaro::Integer [integer](#)

### 12.258.1.2 ModField

```
Givaro::Modular<double> ModField
```

### 12.258.1.3 BasisElement

```
double BasisElement
```

### 12.258.1.4 Element

```
rns_double_elt Element
```

### 12.258.1.5 Element\_ptr

```
rns_double_elt_ptr Element_ptr
```

### 12.258.1.6 ConstElement\_ptr

```
rns_double_elt_cstptr ConstElement_ptr
```

## 12.258.2 Constructor & Destructor Documentation

### 12.258.2.1 rns\_double() [1/4]

```
rns_double (
    const integer & bound,
    size_t pbits,
    bool rnsmod = false,
    long seed = time(NULL)) [inline]
```

### 12.258.2.2 rns\_double() [2/4]

```
rns_double (
    size_t pbits,
    size_t size,
    long seed = time(NULL)) [inline]
```

### 12.258.2.3 rns\_double() [3/4]

```
template<typename Vect >
rns_double (
    const Vect & basis,
    bool rnsmod = false,
    long seed = time(NULL)) [inline]
```

### 12.258.2.4 rns\_double() [4/4]

```
rns_double (
    const RNSIntegerMod< rns_double > & basis,
    bool rnsmod = false,
    long seed = time(NULL)) [inline]
```

## 12.258.3 Member Function Documentation

### 12.258.3.1 precompute\_cst()

```
void precompute_cst (
    size_t K = 0) [inline]
```

### 12.258.3.2 init() [1/3]

```
template<typename T >
void init (
    size_t m,
    size_t n,
    double * Arns,
    size_t rda,
    const T * A,
    size_t lda,
    const integer & maxA,
    bool RNS_MAJOR = false) const [inline]
```

### 12.258.3.3 init() [2/3]

```
void init (
    size_t m,
    size_t n,
    double * Arns,
    size_t rda,
    const integer * A,
    size_t lda,
    size_t k,
    bool RNS_MAJOR = false) const [inline]
```

### 12.258.3.4 init\_transpose()

```
void init_transpose (
    size_t m,
    size_t n,
    double * Arns,
    size_t rda,
    const integer * A,
    size_t lda,
    size_t k,
    bool RNS_MAJOR = false) const [inline]
```

### 12.258.3.5 convert() [1/2]

```
void convert (
    size_t m,
    size_t n,
    integer gamma,
    integer * A,
    size_t lda,
    const double * Arns,
    size_t rda,
    bool RNS_MAJOR = false) const [inline]
```

### 12.258.3.6 convert\_transpose()

```
void convert_transpose (
    size_t m,
    size_t n,
    integer gamma,
    integer * A,
    size_t lda,
    const double * Arns,
    size_t rda,
    bool RNS_MAJOR = false) const [inline]
```

**12.258.3.7 reduce()**

```
void reduce (
    size_t n,
    double * Arns,
    size_t rda,
    bool RNS_MAJOR = false) const [inline]
```

**12.258.3.8 init() [3/3]**

```
template<size_t K>
void init (
    size_t m,
    size_t n,
    double * Arns,
    size_t rda,
    const RecInt::ruint< K > * A,
    size_t lda,
    size_t k,
    bool RNS_MAJOR = false) const [inline]
```

**12.258.3.9 convert() [2/2]**

```
template<size_t K>
void convert (
    size_t m,
    size_t n,
    integer gamma,
    RecInt::ruint< K > * A,
    size_t lda,
    const double * Arns,
    size_t rda,
    integer p = 0,
    bool RNS_MAJOR = false) const [inline]
```

**12.258.4 Field Documentation****12.258.4.1 \_basis**

```
std::vector<double, AlignedAllocator<double, Alignment::CACHE_LINE> > _basis
```

**12.258.4.2 \_basisMax**

```
std::vector<double, AlignedAllocator<double, Alignment::CACHE_LINE> > _basisMax
```

**12.258.4.3 \_negbasis**

```
std::vector<double, AlignedAllocator<double, Alignment::CACHE_LINE> > _negbasis
```

**12.258.4.4 \_invbasis**

```
std::vector<double, AlignedAllocator<double, Alignment::CACHE_LINE> > _invbasis
```

**12.258.4.5 \_field\_rns**

```
std::vector<ModField> _field_rns
```

**12.258.4.6 \_M**

```
integer _M
```



**12.258.4.7 \_Mi**

```
std::vector<integer> _Mi
```

**12.258.4.8 \_MMi**

```
std::vector<double> _MMi
```

**12.258.4.9 \_crt\_in**

```
std::vector<double> _crt_in
```

**12.258.4.10 \_crt\_out**

```
std::vector<double> _crt_out
```

**12.258.4.11 \_size**

```
size_t _size
```

**12.258.4.12 \_pbits**

```
size_t _pbits
```

**12.258.4.13 \_ldm**

```
size_t _ldm
```

**12.258.4.14 \_mi\_sum**

```
integer _mi_sum
```

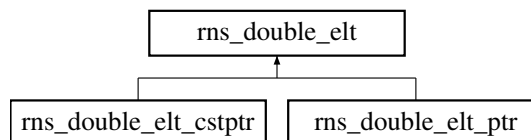
The documentation for this struct was generated from the following files:

- [rns-double.h](#)
- [rns-double-recint.inl](#)
- [rns-double.inl](#)

**12.259 rns\_double\_elt Struct Reference**

```
#include <rns-double-elt.h>
```

Inheritance diagram for rns\_double\_elt:

**Public Member Functions**

- [rns\\_double\\_elt](#) ()
- [~rns\\_double\\_elt](#) ()
- [rns\\_double\\_elt](#) (double \*p, size\_t r, size\_t a=false)
- [rns\\_double\\_elt\\_ptr](#) operator& ()
- [rns\\_double\\_elt\\_cstptr](#) operator& () const
- [rns\\_double\\_elt](#) (const [rns\\_double\\_elt](#) &x)

## Data Fields

- [double \\* \\_ptr](#)
- [size\\_t \\_stride](#)
- [bool \\_alloc](#)

## 12.259.1 Constructor & Destructor Documentation

### 12.259.1.1 `rns_double_elt()` [1/3]

```
rns_double_elt () [inline]
```

### 12.259.1.2 `~rns_double_elt()`

```
~rns_double_elt () [inline]
```

### 12.259.1.3 `rns_double_elt()` [2/3]

```
rns_double_elt (  
    double * p,  
    size_t r,  
    size_t a = false) [inline]
```

### 12.259.1.4 `rns_double_elt()` [3/3]

```
rns_double_elt (  
    const rns\_double\_elt & x) [inline]
```

## 12.259.2 Member Function Documentation

### 12.259.2.1 `operator&()` [1/2]

```
rns\_double\_elt\_ptr operator& () [inline]
```

### 12.259.2.2 `operator&()` [2/2]

```
rns\_double\_elt\_cstptr operator& () const [inline]
```

## 12.259.3 Field Documentation

### 12.259.3.1 `_ptr`

```
double* _ptr
```

### 12.259.3.2 `_stride`

```
size_t _stride
```

### 12.259.3.3 `_alloc`

```
bool _alloc
```

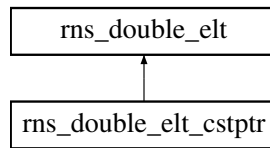
The documentation for this struct was generated from the following file:

- [rns-double-elt.h](#)

## 12.260 rns\_double\_elt\_cstptr Struct Reference

```
#include <rns-double-elt.h>
```

Inheritance diagram for rns\_double\_elt\_cstptr:



### Public Member Functions

- [rns\\_double\\_elt\\_cstptr](#) ()
- [rns\\_double\\_elt\\_cstptr](#) (double \*p, size\_t r)
- [rns\\_double\\_elt\\_cstptr](#) (const [rns\\_double\\_elt\\_ptr](#) &x)
- [rns\\_double\\_elt\\_cstptr](#) (const [rns\\_double\\_elt\\_cstptr](#) &x)
- [rns\\_double\\_elt\\_cstptr](#) ([rns\\_double\\_elt\\_cstptr](#) &&)=default
- [rns\\_double\\_elt\\_cstptr \\* operator&](#) ()
- [rns\\_double\\_elt & operator\\*](#) () const
- [rns\\_double\\_elt operator\[\]](#) (size\_t i) const
- [rns\\_double\\_elt & operator\[\]](#) (size\_t i)
- [rns\\_double\\_elt\\_cstptr operator++](#) ()
- [rns\\_double\\_elt\\_cstptr operator--](#) ()
- [rns\\_double\\_elt\\_cstptr operator+](#) (size\_t inc) const
- [rns\\_double\\_elt\\_cstptr operator-](#) (size\_t inc) const
- [rns\\_double\\_elt\\_cstptr & operator+=](#) (size\_t inc)
- [rns\\_double\\_elt\\_cstptr & operator-=](#) (size\_t inc)
- [rns\\_double\\_elt\\_cstptr & operator=](#) (const [rns\\_double\\_elt\\_cstptr](#) &x)
- [bool operator<](#) (const [rns\\_double\\_elt\\_cstptr](#) &x)
- [bool operator!=](#) (const [rns\\_double\\_elt\\_cstptr](#) &x)
- [rns\\_double\\_elt\\_cstptr operator&](#) () const

### Data Fields

- [rns\\_double\\_elt other](#)
- [double \\* \\_ptr](#)
- [size\\_t \\_stride](#)
- [bool \\_alloc](#)

### 12.260.1 Constructor & Destructor Documentation

#### 12.260.1.1 [rns\\_double\\_elt\\_cstptr](#)() [1/5]

```
rns\_double\_elt\_cstptr () [inline]
```

#### 12.260.1.2 [rns\\_double\\_elt\\_cstptr](#)() [2/5]

```
rns\_double\_elt\_cstptr (
    double * p,
    size_t r) [inline]
```

#### 12.260.1.3 [rns\\_double\\_elt\\_cstptr](#)() [3/5]

```
rns\_double\_elt\_cstptr (
    const rns\_double\_elt\_ptr & x) [inline]
```

**12.260.1.4 rns\_double\_elt\_cstptr() [4/5]**

```
rns_double_elt_cstptr (
    const rns_double_elt_cstptr & x) [inline]
```

**12.260.1.5 rns\_double\_elt\_cstptr() [5/5]**

```
rns_double_elt_cstptr (
    rns_double_elt_cstptr && ) [default]
```

**12.260.2 Member Function Documentation****12.260.2.1 operator&() [1/2]**

```
rns_double_elt_cstptr * operator& () [inline]
```

**12.260.2.2 operator\*()**

```
rns_double_elt & operator* () const [inline]
```

**12.260.2.3 operator[]() [1/2]**

```
rns_double_elt operator[] (
    size_t i) const [inline]
```

**12.260.2.4 operator[]() [2/2]**

```
rns_double_elt & operator[] (
    size_t i) [inline]
```

**12.260.2.5 operator++()**

```
rns_double_elt_cstptr operator++ () [inline]
```

**12.260.2.6 operator--()**

```
rns_double_elt_cstptr operator-- () [inline]
```

**12.260.2.7 operator+()**

```
rns_double_elt_cstptr operator+ (
    size_t inc) const [inline]
```

**12.260.2.8 operator-()**

```
rns_double_elt_cstptr operator- (
    size_t inc) const [inline]
```

**12.260.2.9 operator+=()**

```
rns_double_elt_cstptr & operator+= (
    size_t inc) [inline]
```

**12.260.2.10 operator-=()**

```
rns_double_elt_cstptr & operator-= (
    size_t inc) [inline]
```

**12.260.2.11 operator=()**

```
rns_double_elt_cstptr & operator= (
    const rns_double_elt_cstptr & x) [inline]
```

**12.260.2.12 operator<()**

```
bool operator< (
    const rns_double_elt_cstptr & x) [inline]
```

**12.260.2.13 operator"!=()**

```
bool operator!= (
    const rns_double_elt_cstptr & x) [inline]
```

**12.260.2.14 operator&() [2/2]**

```
rns_double_elt_cstptr operator& () const [inline], [inherited]
```

**12.260.3 Field Documentation****12.260.3.1 other**

```
rns_double_elt other
```

**12.260.3.2 \_ptr**

```
double* _ptr [inherited]
```

**12.260.3.3 \_stride**

```
size_t _stride [inherited]
```

**12.260.3.4 \_alloc**

```
bool _alloc [inherited]
```

The documentation for this struct was generated from the following file:

- [rns-double-elt.h](#)

**12.261 rns\_double\_elt\_ptr Struct Reference**

```
#include <rns-double-elt.h>
```

Inheritance diagram for rns\_double\_elt\_ptr:

**Public Member Functions**

- [rns\\_double\\_elt\\_ptr](#) ()
- [rns\\_double\\_elt\\_ptr](#) (double \*p, size\_t r)
- [rns\\_double\\_elt\\_ptr](#) (const [rns\\_double\\_elt\\_ptr](#) &x)
- [rns\\_double\\_elt\\_ptr](#) (const [rns\\_double\\_elt\\_cstptr](#) &x)
- [rns\\_double\\_elt\\_ptr](#) ([rns\\_double\\_elt\\_ptr](#) &&)=default
- [rns\\_double\\_elt\\_ptr](#) \* [operator&](#) ()

- `rns_double_elt & operator* ()`
- `rns_double_elt operator[] (size_t i) const`
- `rns_double_elt & operator[] (size_t i)`
- `rns_double_elt_ptr operator++ ()`
- `rns_double_elt_ptr operator-- ()`
- `rns_double_elt_ptr operator+ (size_t inc)`
- `rns_double_elt_ptr operator- (size_t inc)`
- `rns_double_elt_ptr & operator+= (size_t inc)`
- `rns_double_elt_ptr & operator-= (size_t inc)`
- `rns_double_elt_ptr & operator= (const rns_double_elt_ptr &x)`
- `bool operator< (const rns_double_elt_ptr &x)`
- `bool operator!= (const rns_double_elt_ptr &x)`
- `rns_double_elt_cstptr operator& () const`

### Data Fields

- `rns_double_elt other`
- `double * _ptr`
- `size_t _stride`
- `bool _alloc`

## 12.261.1 Constructor & Destructor Documentation

### 12.261.1.1 `rns_double_elt_ptr()` [1/5]

```
rns_double_elt_ptr () [inline]
```

### 12.261.1.2 `rns_double_elt_ptr()` [2/5]

```
rns_double_elt_ptr (
    double * p,
    size_t r) [inline]
```

### 12.261.1.3 `rns_double_elt_ptr()` [3/5]

```
rns_double_elt_ptr (
    const rns_double_elt_ptr & x) [inline]
```

### 12.261.1.4 `rns_double_elt_ptr()` [4/5]

```
rns_double_elt_ptr (
    const rns_double_elt_cstptr & x) [inline]
```

### 12.261.1.5 `rns_double_elt_ptr()` [5/5]

```
rns_double_elt_ptr (
    rns_double_elt_ptr && ) [default]
```

## 12.261.2 Member Function Documentation

### 12.261.2.1 `operator&()` [1/2]

```
rns_double_elt_ptr * operator& () [inline]
```

### 12.261.2.2 `operator*()`

```
rns_double_elt & operator* () [inline]
```

**12.261.2.3 operator[]() [1/2]**

```
rns_double_elt operator[] (
    size_t i) const [inline]
```

**12.261.2.4 operator[]() [2/2]**

```
rns_double_elt & operator[] (
    size_t i) [inline]
```

**12.261.2.5 operator++()**

```
rns_double_elt_ptr operator++ () [inline]
```

**12.261.2.6 operator--()**

```
rns_double_elt_ptr operator-- () [inline]
```

**12.261.2.7 operator+()**

```
rns_double_elt_ptr operator+ (
    size_t inc) [inline]
```

**12.261.2.8 operator-()**

```
rns_double_elt_ptr operator- (
    size_t inc) [inline]
```

**12.261.2.9 operator+=()**

```
rns_double_elt_ptr & operator+= (
    size_t inc) [inline]
```

**12.261.2.10 operator-=()**

```
rns_double_elt_ptr & operator-= (
    size_t inc) [inline]
```

**12.261.2.11 operator=()**

```
rns_double_elt_ptr & operator= (
    const rns_double_elt_ptr & x) [inline]
```

**12.261.2.12 operator<()**

```
bool operator< (
    const rns_double_elt_ptr & x) [inline]
```

**12.261.2.13 operator"!=()**

```
bool operator!= (
    const rns_double_elt_ptr & x) [inline]
```

**12.261.2.14 operator&() [2/2]**

```
rns_double_elt_cstptr operator& () const [inline], [inherited]
```

### 12.261.3 Field Documentation

#### 12.261.3.1 other

`rns_double_elt` other

#### 12.261.3.2 \_ptr

`double* _ptr` [inherited]

#### 12.261.3.3 \_stride

`size_t _stride` [inherited]

#### 12.261.3.4 \_alloc

`bool _alloc` [inherited]

The documentation for this struct was generated from the following file:

- [rns-double-elt.h](#)

## 12.262 rns\_double\_extended Struct Reference

```
#include <rns-double.h>
```

### Public Types

- typedef Givaro::Integer [integer](#)
- typedef Givaro::ModularExtended< double > [ModField](#)
- typedef double [BasisElement](#)
- typedef [rns\\_double\\_elt](#) [Element](#)
- typedef [rns\\_double\\_elt\\_ptr](#) [Element\\_ptr](#)
- typedef [rns\\_double\\_elt\\_cstptr](#) [ConstElement\\_ptr](#)

### Public Member Functions

- [rns\\_double\\_extended](#) (const [integer](#) &bound, size\_t pbits, bool rnsmod=false, long seed=time(NULL))
- [rns\\_double\\_extended](#) (size\_t pbits, size\_t size, long seed=time(NULL))
- template<typename Vect >  
  [rns\\_double\\_extended](#) (const Vect &basis, bool rnsmod=false, long seed=time(NULL))
- void [precompute\\_cst](#) ()
- void [init](#) (size\_t m, size\_t n, double \*Arns, size\_t rda, const [integer](#) \*A, size\_t lda, const [integer](#) &maxA, bool RNS\_MAJOR=false) const
- void [init](#) (size\_t m, size\_t n, double \*Arns, size\_t rda, const [integer](#) \*A, size\_t lda, size\_t k, bool RNS\_MAJOR=false)
- void [convert](#) (size\_t m, size\_t n, [integer](#) gamma, [integer](#) \*A, size\_t lda, const double \*Arns, size\_t rda, bool RNS\_MAJOR=false)
- void [init](#) (size\_t m, double \*Arns, const [integer](#) \*A, size\_t lda) const
- void [convert](#) (size\_t m, [integer](#) \*A, const double \*Arns) const
- void [reduce](#) (size\_t n, double \*Arns, size\_t rda, bool RNS\_MAJOR=false) const

### Data Fields

- std::vector< double, AlignedAllocator< double, Alignment::CACHE\_LINE > > [\\_basis](#)
- std::vector< double, AlignedAllocator< double, Alignment::CACHE\_LINE > > [\\_basisMax](#)
- std::vector< double, AlignedAllocator< double, Alignment::CACHE\_LINE > > [\\_negbasis](#)
- std::vector< double, AlignedAllocator< double, Alignment::CACHE\_LINE > > [\\_invbasis](#)
- std::vector< [ModField](#) > [\\_field\\_rns](#)



- [integer\\_M](#)
- `std::vector< integer > _Mi`
- `std::vector< double > _MMi`
- `std::vector< double > _crt_in`
- `std::vector< double > _crt_out`
- `size_t _size`
- `size_t _pbits`
- `size_t _ldm`

## 12.262.1 Member Typedef Documentation

### 12.262.1.1 integer

Givaro::Integer [integer](#)

### 12.262.1.2 ModField

Givaro::ModularExtended<double> [ModField](#)

### 12.262.1.3 BasisElement

double [BasisElement](#)

### 12.262.1.4 Element

[rns\\_double\\_elt](#) Element

### 12.262.1.5 Element\_ptr

[rns\\_double\\_elt\\_ptr](#) Element\_ptr

### 12.262.1.6 ConstElement\_ptr

[rns\\_double\\_elt\\_cstptr](#) ConstElement\_ptr

## 12.262.2 Constructor & Destructor Documentation

### 12.262.2.1 rns\_double\_extended() [1/3]

```
rns\_double\_extended (
    const integer & bound,
    size_t pbits,
    bool rnsmod = false,
    long seed = time(NULL)) [inline]
```

### 12.262.2.2 rns\_double\_extended() [2/3]

```
rns\_double\_extended (
    size_t pbits,
    size_t size,
    long seed = time(NULL)) [inline]
```

### 12.262.2.3 rns\_double\_extended() [3/3]

```
template<typename Vect >
rns\_double\_extended (
    const Vect & basis,
    bool rnsmod = false,
    long seed = time(NULL)) [inline]
```

## 12.262.3 Member Function Documentation

### 12.262.3.1 precompute\_cst()

```
void precompute_cst () [inline]
```

### 12.262.3.2 init() [1/3]

```
void init (
    size_t m,
    size_t n,
    double * Arns,
    size_t rda,
    const integer * A,
    size_t lda,
    const integer & maxA,
    bool RNS_MAJOR = false) const [inline]
```

### 12.262.3.3 init() [2/3]

```
void init (
    size_t m,
    size_t n,
    double * Arns,
    size_t rda,
    const integer * A,
    size_t lda,
    size_t k,
    bool RNS_MAJOR = false) [inline]
```

### 12.262.3.4 convert() [1/2]

```
void convert (
    size_t m,
    size_t n,
    integer gamma,
    integer * A,
    size_t lda,
    const double * Arns,
    size_t rda,
    bool RNS_MAJOR = false) [inline]
```

### 12.262.3.5 init() [3/3]

```
void init (
    size_t m,
    double * Arns,
    const integer * A,
    size_t lda) const [inline]
```

### 12.262.3.6 convert() [2/2]

```
void convert (
    size_t m,
    integer * A,
    const double * Arns) const [inline]
```

### 12.262.3.7 reduce()

```
void reduce (
```

```
    size_t n,  
    double * Arns,  
    size_t rda,  
    bool RNS_MAJOR = false) const [inline]
```

## 12.262.4 Field Documentation

### 12.262.4.1 `_basis`

```
std::vector<double, AlignedAllocator<double, Alignment::CACHE_LINE> > _basis
```

### 12.262.4.2 `_basisMax`

```
std::vector<double, AlignedAllocator<double, Alignment::CACHE_LINE> > _basisMax
```

### 12.262.4.3 `_negbasis`

```
std::vector<double, AlignedAllocator<double, Alignment::CACHE_LINE> > _negbasis
```

### 12.262.4.4 `_invbasis`

```
std::vector<double, AlignedAllocator<double, Alignment::CACHE_LINE> > _invbasis
```

### 12.262.4.5 `_field_rns`

```
std::vector<ModField> _field_rns
```

### 12.262.4.6 `_M`

```
integer _M
```

### 12.262.4.7 `_Mi`

```
std::vector<integer> _Mi
```

### 12.262.4.8 `_MMi`

```
std::vector<double> _MMi
```

### 12.262.4.9 `_crt_in`

```
std::vector<double> _crt_in
```

### 12.262.4.10 `_crt_out`

```
std::vector<double> _crt_out
```

### 12.262.4.11 `_size`

```
size_t _size
```

### 12.262.4.12 `_pbits`

```
size_t _pbits
```

### 12.262.4.13 `_ldm`

```
size_t _ldm
```

The documentation for this struct was generated from the following files:

- [rns-double.h](#)
- [rns-double.inl](#)

## 12.263 RNSElementTag Struct Reference

Representation in a Residue Number System.

```
#include <field-traits.h>
```

### 12.263.1 Detailed Description

Representation in a Residue Number System.

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.264 RNSInteger< RNS > Class Template Reference

```
#include <rns-integer.h>
```

### Data Structures

- class [RandIter](#)

### Public Types

- typedef [RNS::Element](#) [Element](#)
- typedef [RNS::Element\\_ptr](#) [Element\\_ptr](#)
- typedef [RNS::ConstElement\\_ptr](#) [ConstElement\\_ptr](#)

### Public Member Functions

- [RNSInteger](#) (const [RNS](#) &myrns)
- template<typename T >  
  [RNSInteger](#) (const T &F)
- const [RNS](#) & [rns](#) () const
- size\_t [size](#) () const
- bool [isOne](#) (const [Element](#) &x) const
- bool [isMOne](#) (const [Element](#) &x) const
- bool [isZero](#) (const [Element](#) &x) const
- [integer characteristic](#) ([integer](#) &p) const
- [integer cardinality](#) ([integer](#) &p) const
- [Element](#) & [init](#) ([Element](#) &x) const
- [Element](#) & [init](#) ([Element](#) &x, const Givaro::Integer &y) const
- [Element](#) & [reduce](#) ([Element](#) &x, const [Element](#) &y) const
- [Element](#) & [reduce](#) ([Element](#) &x) const
- Givaro::Integer [convert](#) (Givaro::Integer &x, const [Element](#) &y) const
- [Element](#) & [assign](#) ([Element](#) &x, const [Element](#) &y) const
- std::ostream & [write](#) (std::ostream &os, const [Element](#) &y) const
- std::ostream & [write](#) (std::ostream &os) const

### Data Fields

- [Element one](#)
- [Element mOne](#)
- [Element zero](#)

### Protected Types

- typedef [RNS::BasisElement](#) [BasisElement](#)
- typedef Givaro::Integer [integer](#)

**Protected Attributes**

- const [RNS](#) \* [\\_rns](#)

**12.264.1 Member Typedef Documentation****12.264.1.1 BasisElement**

```
template<typename RNS >
RNS::BasisElement BasisElement [protected]
```

**12.264.1.2 integer**

```
template<typename RNS >
Givaro::Integer integer [protected]
```

**12.264.1.3 Element**

```
template<typename RNS >
RNS::Element Element
```

**12.264.1.4 Element\_ptr**

```
template<typename RNS >
RNS::Element\_ptr Element\_ptr
```

**12.264.1.5 ConstElement\_ptr**

```
template<typename RNS >
RNS::ConstElement\_ptr ConstElement\_ptr
```

**12.264.2 Constructor & Destructor Documentation****12.264.2.1 RNSInteger() [1/2]**

```
template<typename RNS >
RNSInteger (
    const RNS & myrns) [inline]
```

**12.264.2.2 RNSInteger() [2/2]**

```
template<typename RNS >
template<typename T >
RNSInteger (
    const T & F) [inline]
```

**12.264.3 Member Function Documentation****12.264.3.1 rns()**

```
template<typename RNS >
const RNS & rns () const [inline]
```

**12.264.3.2 size()**

```
template<typename RNS >
size_t size () const [inline]
```

**12.264.3.3 isOne()**

```
template<typename RNS >
bool isOne (
    const Element & x) const [inline]
```

**12.264.3.4 isMOne()**

```
template<typename RNS >
bool isMOne (
    const Element & x) const [inline]
```

**12.264.3.5 isZero()**

```
template<typename RNS >
bool isZero (
    const Element & x) const [inline]
```

**12.264.3.6 characteristic()**

```
template<typename RNS >
integer characteristic (
    integer & p) const [inline]
```

**12.264.3.7 cardinality()**

```
template<typename RNS >
integer cardinality (
    integer & p) const [inline]
```

**12.264.3.8 init() [1/2]**

```
template<typename RNS >
Element & init (
    Element & x) const [inline]
```

**12.264.3.9 init() [2/2]**

```
template<typename RNS >
Element & init (
    Element & x,
    const Givaro::Integer & y) const [inline]
```

**12.264.3.10 reduce() [1/2]**

```
template<typename RNS >
Element & reduce (
    Element & x,
    const Element & y) const [inline]
```

**12.264.3.11 reduce() [2/2]**

```
template<typename RNS >
Element & reduce (
    Element & x) const [inline]
```

#### 12.264.3.12 convert()

```
template<typename RNS >
Givaro::Integer convert (
    Givaro::Integer & x,
    const Element & y) const [inline]
```

#### 12.264.3.13 assign()

```
template<typename RNS >
Element & assign (
    Element & x,
    const Element & y) const [inline]
```

#### 12.264.3.14 write() [1/2]

```
template<typename RNS >
std::ostream & write (
    std::ostream & os,
    const Element & y) const [inline]
```

#### 12.264.3.15 write() [2/2]

```
template<typename RNS >
std::ostream & write (
    std::ostream & os) const [inline]
```

### 12.264.4 Field Documentation

#### 12.264.4.1 \_rns

```
template<typename RNS >
const RNS* _rns [protected]
```

#### 12.264.4.2 one

```
template<typename RNS >
Element one
```

#### 12.264.4.3 mOne

```
template<typename RNS >
Element mOne
```

#### 12.264.4.4 zero

```
template<typename RNS >
Element zero
```

The documentation for this class was generated from the following files:

- [field-traits.h](#)
- [rns-integer.h](#)
- [rns.h](#)

## 12.265 RNSIntegerMod< RNS > Class Template Reference

```
#include <rns-integer-mod.h>
```

## Data Structures

- class [RandIter](#)

## Public Types

- typedef [RNS::Element](#) [Element](#)
- typedef [RNS::Element\\_ptr](#) [Element\\_ptr](#)
- typedef [RNS::ConstElement\\_ptr](#) [ConstElement\\_ptr](#)

## Public Member Functions

- [RNSIntegerMod](#) (const [integer](#) &p, const [RNS](#) &myrns)
- const [rns\\_double](#) & [rns](#) () const
- const [RNSInteger](#)< [RNS](#) > & [delayed](#) () const
- [size\\_t](#) [size](#) () const
- bool [isOne](#) (const [Element](#) &x) const
- bool [isMOne](#) (const [Element](#) &x) const
- bool [isZero](#) (const [Element](#) &x) const
- [integer](#) & [characteristic](#) ([integer](#) &p) const
- [integer](#) [characteristic](#) () const
- [integer](#) & [cardinality](#) ([integer](#) &p) const
- [integer](#) [cardinality](#) () const
- [integer](#) [minElement](#) () const
- [integer](#) [maxElement](#) () const
- [Element](#) & [init](#) ([Element](#) &x) const
- [Element](#) & [init](#) ([Element](#) &x, const [Givaro::Integer](#) &y) const
- [Element](#) & [reduce](#) ([Element](#) &x, const [Element](#) &y) const
- [Element](#) & [reduce](#) ([Element](#) &x) const
- [Element](#) & [init](#) ([Element](#) &x, const [Element](#) &y) const
- [Givaro::Integer](#) [convert](#) ([Givaro::Integer](#) &x, const [Element](#) &y) const
- [Element](#) & [assign](#) ([Element](#) &x, const [Element](#) &y) const
- [Element](#) & [add](#) ([Element](#) &x, const [Element](#) &y, const [Element](#) &z) const
- [Element](#) & [sub](#) ([Element](#) &x, const [Element](#) &y, const [Element](#) &z) const
- [Element](#) & [neg](#) ([Element](#) &x, const [Element](#) &y) const
- [Element](#) & [mul](#) ([Element](#) &x, const [Element](#) &y, const [Element](#) &z) const
- [Element](#) & [axpyin](#) ([Element](#) &x, const [Element](#) &y, const [Element](#) &z) const
- [Element](#) & [inv](#) ([Element](#) &x, const [Element](#) &y) const
- bool [areEqual](#) (const [Element](#) &x, const [Element](#) &y) const
- [std::ostream](#) & [write](#) ([std::ostream](#) &os, const [Element](#) &y) const
- [std::ostream](#) & [write](#) ([std::ostream](#) &os) const
- void [reduce\\_modp](#) ([size\\_t](#) n, [Element\\_ptr](#) B) const
- [std::ostream](#) & [write\\_matrix](#) ([std::ostream](#) &c, const double \*E, int n, int m, int lda) const
- [std::ostream](#) & [write\\_matrix\\_long](#) ([std::ostream](#) &c, const double \*E, int n, int m, int lda) const
- void [reduce\\_modp](#) ([size\\_t](#) m, [size\\_t](#) n, [Element\\_ptr](#) B, [size\\_t](#) lda) const
- void [reduce\\_modp\\_rnsmajor](#) ([size\\_t](#) n, [Element\\_ptr](#) B) const

## Data Fields

- [Element](#) one
- [Element](#) mOne
- [Element](#) zero

## Protected Types

- typedef [RNS::BasisElement](#) [BasisElement](#)
- typedef [Givaro::Modular](#)< [BasisElement](#) > [ModField](#)
- typedef [Givaro::Integer](#) [integer](#)



**Protected Attributes**

- [integer\\_p](#)
- `std::vector< BasisElement, AlignedAllocator< BasisElement, Alignment::CACHE_LINE > > \_Mi\_modp\_rns`
- `std::vector< BasisElement, AlignedAllocator< BasisElement, Alignment::CACHE_LINE > > \_iM\_modp\_rns`
- `const RNS * \_rns`
- `Givaro::Modular< Givaro::Integer > \_F`
- [RNSInteger< RNS > \[\\\_RNSdelayed\]\(#\)](#)

**12.265.1 Member Typedef Documentation****12.265.1.1 Element**

```
template<typename RNS >
RNS::Element Element
```

**12.265.1.2 Element\_ptr**

```
template<typename RNS >
RNS::Element\_ptr Element\_ptr
```

**12.265.1.3 ConstElement\_ptr**

```
template<typename RNS >
RNS::ConstElement\_ptr ConstElement\_ptr
```

**12.265.1.4 BasisElement**

```
template<typename RNS >
RNS::BasisElement BasisElement [protected]
```

**12.265.1.5 ModField**

```
template<typename RNS >
Givaro::Modular<BasisElement> ModField [protected]
```

**12.265.1.6 integer**

```
template<typename RNS >
Givaro::Integer integer [protected]
```

**12.265.2 Constructor & Destructor Documentation****12.265.2.1 RNSIntegerMod()**

```
template<typename RNS >
RNSIntegerMod (
    const integer & p,
    const RNS & myrns) [inline]
```

**12.265.3 Member Function Documentation****12.265.3.1 rns()**

```
template<typename RNS >
const rns\_double & rns () const [inline]
```

**12.265.3.2 delayed()**

```
template<typename RNS >
const RNSInteger<RNS> & delayed () const [inline]
```

**12.265.3.3 size()**

```
template<typename RNS >
size_t size () const [inline]
```

**12.265.3.4 isOne()**

```
template<typename RNS >
bool isOne (
    const Element & x) const [inline]
```

**12.265.3.5 isMOne()**

```
template<typename RNS >
bool isMOne (
    const Element & x) const [inline]
```

**12.265.3.6 isZero()**

```
template<typename RNS >
bool isZero (
    const Element & x) const [inline]
```

**12.265.3.7 characteristic() [1/2]**

```
template<typename RNS >
integer & characteristic (
    integer & p) const [inline]
```

**12.265.3.8 characteristic() [2/2]**

```
template<typename RNS >
integer characteristic () const [inline]
```

**12.265.3.9 cardinality() [1/2]**

```
template<typename RNS >
integer & cardinality (
    integer & p) const [inline]
```

**12.265.3.10 cardinality() [2/2]**

```
template<typename RNS >
integer cardinality () const [inline]
```

**12.265.3.11 minElement()**

```
template<typename RNS >
integer minElement () const [inline]
```

**12.265.3.12 maxElement()**

```
template<typename RNS >
integer maxElement () const [inline]
```

**12.265.3.13 init() [1/3]**

```
template<typename RNS >
Element & init (
    Element & x) const [inline]
```

**12.265.3.14 init() [2/3]**

```
template<typename RNS >
Element & init (
    Element & x,
    const Givaro::Integer & y) const [inline]
```

**12.265.3.15 reduce() [1/2]**

```
template<typename RNS >
Element & reduce (
    Element & x,
    const Element & y) const [inline]
```

**12.265.3.16 reduce() [2/2]**

```
template<typename RNS >
Element & reduce (
    Element & x) const [inline]
```

**12.265.3.17 init() [3/3]**

```
template<typename RNS >
Element & init (
    Element & x,
    const Element & y) const [inline]
```

**12.265.3.18 convert()**

```
template<typename RNS >
Givaro::Integer convert (
    Givaro::Integer & x,
    const Element & y) const [inline]
```

**12.265.3.19 assign()**

```
template<typename RNS >
Element & assign (
    Element & x,
    const Element & y) const [inline]
```

**12.265.3.20 add()**

```
template<typename RNS >
Element & add (
    Element & x,
    const Element & y,
    const Element & z) const [inline]
```

**12.265.3.21 sub()**

```
template<typename RNS >
Element & sub (
    Element & x,
    const Element & y,
    const Element & z) const [inline]
```

**12.265.3.22 neg()**

```
template<typename RNS >
Element & neg (
    Element & x,
    const Element & y) const [inline]
```

**12.265.3.23 mul()**

```
template<typename RNS >
Element & mul (
    Element & x,
    const Element & y,
    const Element & z) const [inline]
```

**12.265.3.24 axpyin()**

```
template<typename RNS >
Element & axpyin (
    Element & x,
    const Element & y,
    const Element & z) const [inline]
```

**12.265.3.25 inv()**

```
template<typename RNS >
Element & inv (
    Element & x,
    const Element & y) const [inline]
```

**12.265.3.26 areEqual()**

```
template<typename RNS >
bool areEqual (
    const Element & x,
    const Element & y) const [inline]
```

**12.265.3.27 write() [1/2]**

```
template<typename RNS >
std::ostream & write (
    std::ostream & os,
    const Element & y) const [inline]
```

**12.265.3.28 write() [2/2]**

```
template<typename RNS >
std::ostream & write (
    std::ostream & os) const [inline]
```

**12.265.3.29 reduce\_modp() [1/2]**

```
template<typename RNS >
void reduce_modp (
    size_t n,
    Element_ptr B) const [inline]
```

**12.265.3.30 write\_matrix()**

```
template<typename RNS >
std::ostream & write_matrix (
    std::ostream & c,
    const double * E,
    int n,
    int m,
    int lda) const [inline]
```

**12.265.3.31 write\_matrix\_long()**

```
template<typename RNS >
std::ostream & write_matrix_long (
    std::ostream & c,
    const double * E,
    int n,
    int m,
    int lda) const [inline]
```

**12.265.3.32 reduce\_modp() [2/2]**

```
template<typename RNS >
void reduce_modp (
    size_t m,
    size_t n,
    Element_ptr B,
    size_t lda) const [inline]
```

**12.265.3.33 reduce\_modp\_rnsmajor()**

```
template<typename RNS >
void reduce_modp_rnsmajor (
    size_t n,
    Element_ptr B) const [inline]
```

**12.265.4 Field Documentation****12.265.4.1 \_p**

```
template<typename RNS >
integer _p [protected]
```

**12.265.4.2 \_Mi\_modp\_rns**

```
template<typename RNS >
std::vector<BasisElement, AlignedAllocator<BasisElement, Alignment::CACHE_LINE> > _Mi_modp_↔
rns [protected]
```

**12.265.4.3 \_iM\_modp\_rns**

```
template<typename RNS >
std::vector<BasisElement, AlignedAllocator<BasisElement, Alignment::CACHE_LINE> > _iM_modp_↔
rns [protected]
```

**12.265.4.4 \_rns**

```
template<typename RNS >
const RNS* _rns [protected]
```

**12.265.4.5 \_F**

```
template<typename RNS >
Givaro::Modular<Givaro::Integer> _F [protected]
```

**12.265.4.6 \_RNSdelayed**

```
template<typename RNS >
RNSInteger<RNS> _RNSdelayed [protected]
```

**12.265.4.7 one**

```
template<typename RNS >
Element one
```

**12.265.4.8 mOne**

```
template<typename RNS >
Element mOne
```

**12.265.4.9 zero**

```
template<typename RNS >
Element zero
```

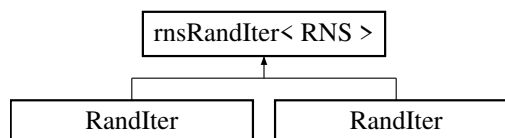
The documentation for this class was generated from the following files:

- [field-traits.h](#)
- [rns-integer-mod.h](#)
- [rns.h](#)

**12.266 rnsRandIter< RNS > Class Template Reference**

```
#include <rns-double.h>
```

Inheritance diagram for rnsRandIter< RNS >:

**Public Member Functions**

- [rnsRandIter](#) (const [RNS](#) &R, uint64\_t seed=0)
- [RNS::Element & random](#) (typename [RNS::Element](#) &elt) const  
*RNS ring Element random assignement.*
- [RNS::Element & operator\(\)](#) (typename [RNS::Element](#) &elt) const
- [RNS::Element operator\(\)](#) () const
- [RNS::Element random](#) () const
- const [RNS](#) & [ring](#) () const

**12.266.1 Constructor & Destructor Documentation****12.266.1.1 rnsRandIter()**

```
template<typename RNS >
rnsRandIter (
    const RNS & R,
    uint64_t seed = 0) [inline]
```

## 12.266.2 Member Function Documentation

### 12.266.2.1 random() [1/2]

```
template<typename RNS >
RNS::Element & random (
    typename RNS::Element & elt) const [inline]
```

RNS ring Element random assignment.

Element is supposed to be initialized

#### Returns

random ring Element

### 12.266.2.2 operator>() [1/2]

```
template<typename RNS >
RNS::Element & operator() (
    typename RNS::Element & elt) const [inline]
```

### 12.266.2.3 operator>() [2/2]

```
template<typename RNS >
RNS::Element operator() () const [inline]
```

### 12.266.2.4 random() [2/2]

```
template<typename RNS >
RNS::Element random () const [inline]
```

### 12.266.2.5 ring()

```
template<typename RNS >
const RNS & ring () const [inline]
```

The documentation for this class was generated from the following file:

- [rns-double.h](#)

## 12.267 Row Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.268 ruint< K > Class Template Reference

The documentation for this class was generated from the following file:

- [field-traits.h](#)

## 12.269 ScalFunctions< Element > Struct Template Reference

Inheritance diagram for ScalFunctions< Element >:



## Public Types

- using `vectElt` = `vector<Element>`

## Static Public Member Functions

- static void `genInputs` (`vector< vectElt > &inputs`)
- static void `genInputsWithZero` (`vector< vectElt > &inputs`)
- static Element `zero` ()
- static Element `vand` (Element x1, Element x2)
- static Element `vor` (Element x1, Element x2)
- static Element `vxor` (Element x1, Element x2)
- static Element `vandnot` (Element x1, Element x2)
- static Element `add` (Element x1, Element x2)
- static Element `addin` (Element &x1, Element x2)
- static Element `sub` (Element x1, Element x2)
- static Element `subin` (Element &x1, Element x2)
- static Element `mul` (Element x1, Element x2)
- static Element `mulin` (Element &x1, Element x2)
- static Element `div` (Element x1, Element x2)
- static Element `fmadd` (Element x1, Element x2, Element x3)
- static Element `fmaddin` (Element &x1, Element x2, Element x3)
- static Element `fmsub` (Element x1, Element x2, Element x3)
- static Element `fmsubin` (Element &x1, Element x2, Element x3)
- static Element `fnmadd` (Element x1, Element x2, Element x3)
- static Element `fnmaddin` (Element &x1, Element x2, Element x3)
- static Element `lesser` (Element x1, Element x2)
- static Element `lesser_eq` (Element x1, Element x2)
- static Element `greater` (Element x1, Element x2)
- static Element `greater_eq` (Element x1, Element x2)
- static Element `eq` (Element x1, Element x2)
- static `vectElt` `unpacklo` (`vectElt` a, `vectElt` b)
- static `vectElt` `unpackhi` (`vectElt` a, `vectElt` b)
- static void `unpacklohi` (`vectElt` &lo, `vectElt` &hi, `vectElt` a, `vectElt` b)
- static `vectElt` `pack_even` (`vectElt` a, `vectElt` b)
- static `vectElt` `pack_odd` (`vectElt` a, `vectElt` b)
- static void `pack` (`vectElt` &even, `vectElt` &odd, `vectElt` a, `vectElt` b)
- `template<uint16_t s>`  
static `vectElt` `blend` (`vectElt` a, `vectElt` b)

## 12.269.1 Member Typedef Documentation

### 12.269.1.1 vectElt

```
template<typename Element >
using vectElt = vector<Element>
```

## 12.269.2 Member Function Documentation

### 12.269.2.1 genInputs()

```
template<typename Element >
static void genInputs (
    vector< vectElt > & inputs) [inline], [static]
```



### 12.269.2.2 genInputsWithZero()

```
template<typename Element >
static void genInputsWithZero (
    vector< vectElt > & inputs) [inline], [static]
```

### 12.269.2.3 zero()

```
template<typename Element >
static Element zero () [inline], [static]
```

### 12.269.2.4 vand()

```
template<typename Element >
static Element vand (
    Element x1,
    Element x2) [inline], [static]
```

### 12.269.2.5 vor()

```
template<typename Element >
static Element vor (
    Element x1,
    Element x2) [inline], [static]
```

### 12.269.2.6 vxor()

```
template<typename Element >
static Element vxor (
    Element x1,
    Element x2) [inline], [static]
```

### 12.269.2.7 vandnot()

```
template<typename Element >
static Element vandnot (
    Element x1,
    Element x2) [inline], [static]
```

### 12.269.2.8 add()

```
template<typename Element >
static Element add (
    Element x1,
    Element x2) [inline], [static]
```

### 12.269.2.9 addin()

```
template<typename Element >
static Element addin (
    Element & x1,
    Element x2) [inline], [static]
```

### 12.269.2.10 sub()

```
template<typename Element >
static Element sub (
    Element x1,
    Element x2) [inline], [static]
```

**12.269.2.11 subin()**

```
template<typename Element >
static Element subin (
    Element & x1,
    Element x2) [inline], [static]
```

**12.269.2.12 mul()**

```
template<typename Element >
static Element mul (
    Element x1,
    Element x2) [inline], [static]
```

**12.269.2.13 mulin()**

```
template<typename Element >
static Element mulin (
    Element & x1,
    Element x2) [inline], [static]
```

**12.269.2.14 div()**

```
template<typename Element >
static Element div (
    Element x1,
    Element x2) [inline], [static]
```

**12.269.2.15 fmadd()**

```
template<typename Element >
static Element fmadd (
    Element x1,
    Element x2,
    Element x3) [inline], [static]
```

**12.269.2.16 fmaddin()**

```
template<typename Element >
static Element fmaddin (
    Element & x1,
    Element x2,
    Element x3) [inline], [static]
```

**12.269.2.17 fmsub()**

```
template<typename Element >
static Element fmsub (
    Element x1,
    Element x2,
    Element x3) [inline], [static]
```

**12.269.2.18 fmsubin()**

```
template<typename Element >
static Element fmsubin (
    Element & x1,
    Element x2,
    Element x3) [inline], [static]
```

**12.269.2.19 fnmadd()**

```
template<typename Element >
static Element fnmadd (
    Element x1,
    Element x2,
    Element x3) [inline], [static]
```

**12.269.2.20 fnmaddin()**

```
template<typename Element >
static Element fnmaddin (
    Element & x1,
    Element x2,
    Element x3) [inline], [static]
```

**12.269.2.21 lesser()**

```
template<typename Element >
static Element lesser (
    Element x1,
    Element x2) [inline], [static]
```

**12.269.2.22 lesser\_eq()**

```
template<typename Element >
static Element lesser_eq (
    Element x1,
    Element x2) [inline], [static]
```

**12.269.2.23 greater()**

```
template<typename Element >
static Element greater (
    Element x1,
    Element x2) [inline], [static]
```

**12.269.2.24 greater\_eq()**

```
template<typename Element >
static Element greater_eq (
    Element x1,
    Element x2) [inline], [static]
```

**12.269.2.25 eq()**

```
template<typename Element >
static Element eq (
    Element x1,
    Element x2) [inline], [static]
```

**12.269.2.26 unpacklo()**

```
template<typename Element >
static vectElt unpacklo (
    vectElt a,
    vectElt b) [inline], [static]
```

**12.269.2.27 unpackhi()**

```
template<typename Element >
static vectElt unpackhi (
    vectElt a,
    vectElt b) [inline], [static]
```

**12.269.2.28 unpacklohi()**

```
template<typename Element >
static void unpacklohi (
    vectElt & lo,
    vectElt & hi,
    vectElt a,
    vectElt b) [inline], [static]
```

**12.269.2.29 pack\_even()**

```
template<typename Element >
static vectElt pack_even (
    vectElt a,
    vectElt b) [inline], [static]
```

**12.269.2.30 pack\_odd()**

```
template<typename Element >
static vectElt pack_odd (
    vectElt a,
    vectElt b) [inline], [static]
```

**12.269.2.31 pack()**

```
template<typename Element >
static void pack (
    vectElt & even,
    vectElt & odd,
    vectElt a,
    vectElt b) [inline], [static]
```

**12.269.2.32 blend()**

```
template<typename Element >
template<uint16_t s>
static vectElt blend (
    vectElt a,
    vectElt b) [inline], [static]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.270 ScalFunctionsBase< Element, Enable > Struct Template Reference

Inheritance diagram for ScalFunctionsBase< Element, Enable >:



The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.271 ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type > Struct Template Reference

### Data Structures

- class [FloatingPointTestDistribution](#)

### Static Public Member Functions

- static [FloatingPointTestDistribution](#) [get\\_default\\_random\\_generator](#) ()
- static Element [ceil](#) (Element x)
- static Element [floor](#) (Element x)
- static Element [round](#) (Element x)
- static Element [blendv](#) (Element a, Element b, Element mask)
- static Element [fma](#) (Element x, Element y, Element z)

### Static Public Attributes

- static constexpr Element [\\_zero](#) = 0.0
- static constexpr Element [cmp\\_true](#) = NAN
- static constexpr Element [cmp\\_false](#) = \_zero

## 12.271.1 Member Function Documentation

### 12.271.1.1 [get\\_default\\_random\\_generator\(\)](#)

```
template<typename Element >
static FloatingPointTestDistribution get_default_random_generator () [inline], [static]
```

### 12.271.1.2 [ceil\(\)](#)

```
template<typename Element >
static Element ceil (
    Element x) [inline], [static]
```

### 12.271.1.3 [floor\(\)](#)

```
template<typename Element >
static Element floor (
    Element x) [inline], [static]
```

### 12.271.1.4 [round\(\)](#)

```
template<typename Element >
static Element round (
    Element x) [inline], [static]
```

### 12.271.1.5 blendv()

```
template<typename Element >
static Element blendv (
    Element a,
    Element b,
    Element mask) [inline], [static]
```

### 12.271.1.6 fma()

```
template<typename Element >
static Element fma (
    Element x,
    Element y,
    Element z) [inline], [static]
```

## 12.271.2 Field Documentation

### 12.271.2.1 \_zero

```
template<typename Element >
Element _zero = 0.0 [static], [constexpr]
```

### 12.271.2.2 cmp\_true

```
template<typename Element >
Element cmp_true = NAN [static], [constexpr]
```

### 12.271.2.3 cmp\_false

```
template<typename Element >
Element cmp_false = _zero [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.272 ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type > Struct Template Reference

### Static Public Member Functions

- static uniform\_int\_distribution< Element > [get\\_default\\_random\\_generator](#) ()
- static Element [round](#) (Element x)
- static Element [fma](#) (Element x, Element y, Element z)
- static Element [mullo](#) (Element x1, Element x2)
- static Element [mulhi](#) (Element x1, Element x2)
- static Element [mulx](#) (Element x1, Element x2)
- static Element [fmaddx](#) (Element x1, Element x2, Element x3)
- static Element [fmaddxin](#) (Element &x1, Element x2, Element x3)
- static Element [fmsubx](#) (Element x1, Element x2, Element x3)
- static Element [fmsubxin](#) (Element &x1, Element x2, Element x3)
- static Element [fnmaddx](#) (Element x1, Element x2, Element x3)
- static Element [fnmaddxin](#) (Element &x1, Element x2, Element x3)
- template<int s>
 static Element [sra](#) (Element x1)
- template<int s>
 static Element [srl](#) (Element x1)
- template<int s>
 static Element [sll](#) (Element x1)

### Static Public Attributes

- static constexpr Element [\\_zero](#) = 0
- static constexpr Element [cmp\\_true](#) = -1
- static constexpr Element [cmp\\_false](#) = [\\_zero](#)

## 12.272.1 Member Function Documentation

### 12.272.1.1 [get\\_default\\_random\\_generator\(\)](#)

```
template<typename Element >  
static uniform_int_distribution< Element > get_default_random_generator () [inline], [static]
```

### 12.272.1.2 [round\(\)](#)

```
template<typename Element >  
static Element round (  
    Element x) [inline], [static]
```

### 12.272.1.3 [fma\(\)](#)

```
template<typename Element >  
static Element fma (  
    Element x,  
    Element y,  
    Element z) [inline], [static]
```

### 12.272.1.4 [mullo\(\)](#)

```
template<typename Element >  
static Element mullo (  
    Element x1,  
    Element x2) [inline], [static]
```

### 12.272.1.5 [mulhi\(\)](#)

```
template<typename Element >  
static Element mulhi (  
    Element x1,  
    Element x2) [inline], [static]
```

### 12.272.1.6 [mulx\(\)](#)

```
template<typename Element >  
static Element mulx (  
    Element x1,  
    Element x2) [inline], [static]
```

### 12.272.1.7 [fmaddx\(\)](#)

```
template<typename Element >  
static Element fmaddx (  
    Element x1,  
    Element x2,  
    Element x3) [inline], [static]
```

**12.272.1.8 fmaddxin()**

```
template<typename Element >
static Element fmaddxin (
    Element & x1,
    Element x2,
    Element x3) [inline], [static]
```

**12.272.1.9 fmsubx()**

```
template<typename Element >
static Element fmsubx (
    Element x1,
    Element x2,
    Element x3) [inline], [static]
```

**12.272.1.10 fmsubxin()**

```
template<typename Element >
static Element fmsubxin (
    Element & x1,
    Element x2,
    Element x3) [inline], [static]
```

**12.272.1.11 fnmaddx()**

```
template<typename Element >
static Element fnmaddx (
    Element x1,
    Element x2,
    Element x3) [inline], [static]
```

**12.272.1.12 fnmaddxin()**

```
template<typename Element >
static Element fnmaddxin (
    Element & x1,
    Element x2,
    Element x3) [inline], [static]
```

**12.272.1.13 sra()**

```
template<typename Element >
template<int s>
static Element sra (
    Element x1) [inline], [static]
```

**12.272.1.14 srl()**

```
template<typename Element >
template<int s>
static Element srl (
    Element x1) [inline], [static]
```

**12.272.1.15 sll()**

```
template<typename Element >
template<int s>
static Element sll (
    Element x1) [inline], [static]
```



## 12.272.2 Field Documentation

### 12.272.2.1 `_zero`

```
template<typename Element >
Element _zero = 0 [static], [constexpr]
```

### 12.272.2.2 `cmp_true`

```
template<typename Element >
Element cmp_true = -1 [static], [constexpr]
```

### 12.272.2.3 `cmp_false`

```
template<typename Element >
Element cmp_false = _zero [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.273 Sequential Struct Reference

### Public Member Functions

- [Sequential](#) ()
- [Sequential](#) (size\_t nth)
- template<class Cut , class Param >  
[Sequential](#) ([Parallel](#)< Cut, Param > &)
- size\_t [numthreads](#) () const

### Friends

- std::ostream & [operator<<](#) (std::ostream &out, const [Sequential](#) &)

## 12.273.1 Constructor & Destructor Documentation

### 12.273.1.1 `Sequential()` [1/3]

```
Sequential () [inline]
```

### 12.273.1.2 `Sequential()` [2/3]

```
Sequential (
    size_t nth) [inline]
```

### 12.273.1.3 `Sequential()` [3/3]

```
template<class Cut , class Param >
Sequential (
    Parallel< Cut, Param > & ) [inline]
```

## 12.273.2 Member Function Documentation

### 12.273.2.1 `numthreads()`

```
size_t numthreads () const [inline]
```

### 12.273.3 Friends And Related Symbol Documentation

#### 12.273.3.1 operator<<

```
std::ostream & operator<< (
    std::ostream & out,
    const Sequential & ) [friend]
```

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

### 12.274 Simd128\_impl< ArithType, Int, Signed, Size > Struct Template Reference

The documentation for this struct was generated from the following file:

- [simd128.inl](#)

### 12.275 Simd128\_impl< true, false, true, 4 > Struct Reference

The documentation for this struct was generated from the following file:

- [simd128\\_float.inl](#)

### 12.276 Simd128\_impl< true, false, true, 8 > Struct Reference

The documentation for this struct was generated from the following file:

- [simd128\\_double.inl](#)

### 12.277 Simd128\_impl< true, true, false, 2 > Struct Reference

Inheritance diagram for Simd128\_impl< true, true, false, 2 >:



#### Data Structures

- union [Converter](#)

#### Public Types

- using [scalar\\_t](#) = uint16\_t
- using [aligned\\_allocator](#) = AlignedAllocator<[scalar\\_t](#), Alignment([alignment](#))>
- using [aligned\\_vector](#) = std::vector<[scalar\\_t](#), [aligned\\_allocator](#)>
- template<class [Field](#) >  
using [is\\_same\\_element](#) = std::is\_same<typename [Field::Element](#), [scalar\\_t](#)>
- using [vect\\_t](#) = \_\_m128i

## Static Public Member Functions

- static const std::string [type\\_string](#) ()
- static [INLINE CONST vect\\_t set1](#) (const [scalar\\_t](#) x)
- static [INLINE CONST vect\\_t set](#) (const [scalar\\_t](#) x0, const [scalar\\_t](#) x1, const [scalar\\_t](#) x2, const [scalar\\_t](#) x3, const [scalar\\_t](#) x4, const [scalar\\_t](#) x5, const [scalar\\_t](#) x6, const [scalar\\_t](#) x7)
- template<class T >  
static [INLINE PURE vect\\_t gather](#) (const [scalar\\_t](#) \*const p, const T \*const idx)
- static [INLINE PURE vect\\_t load](#) (const [scalar\\_t](#) \*const p)
- static [INLINE PURE vect\\_t loadu](#) (const [scalar\\_t](#) \*const p)
- static [INLINE void store](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)
- static [INLINE void storeu](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)
- static [INLINE void stream](#) ([scalar\\_t](#) \*p, const [vect\\_t](#) v)
- template<int s>  
static [INLINE CONST vect\\_t sra](#) (const [vect\\_t](#) a)
- static [INLINE CONST vect\\_t greater](#) ([vect\\_t](#) a, [vect\\_t](#) b)
- static [INLINE CONST vect\\_t lesser](#) ([vect\\_t](#) a, [vect\\_t](#) b)
- static [INLINE CONST vect\\_t greater\\_eq](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t lesser\\_eq](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t mulhi](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t mulx](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t fmaddx](#) (const [vect\\_t](#) c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE vect\\_t fmaddxin](#) ([vect\\_t](#) &c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t fnmaddx](#) (const [vect\\_t](#) c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE vect\\_t fnmaddxin](#) ([vect\\_t](#) &c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t fmsubx](#) (const [vect\\_t](#) c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE vect\\_t fmsubxin](#) ([vect\\_t](#) &c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST scalar\\_t hadd\\_to\\_scal](#) (const [vect\\_t](#) a)
- template<class T >  
static constexpr bool [valid](#) (T \*p)
- template<class T >  
static constexpr bool [compliant](#) (T n)
- static [INLINE CONST vect\\_t set1](#) (const [scalar\\_t](#) x)
- static [INLINE CONST vect\\_t set](#) (const [scalar\\_t](#) x0, const [scalar\\_t](#) x1, const [scalar\\_t](#) x2, const [scalar\\_t](#) x3, const [scalar\\_t](#) x4, const [scalar\\_t](#) x5, const [scalar\\_t](#) x6, const [scalar\\_t](#) x7)
- template<class T >  
static [INLINE PURE vect\\_t gather](#) (const [scalar\\_t](#) \*const p, const T \*const idx)
- static [INLINE PURE vect\\_t load](#) (const [scalar\\_t](#) \*const p)
- static [INLINE PURE vect\\_t loadu](#) (const [scalar\\_t](#) \*const p)
- static [INLINE void store](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)
- static [INLINE void storeu](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)
- static [INLINE void stream](#) ([scalar\\_t](#) \*p, const [vect\\_t](#) v)
- template<int s>  
static [INLINE CONST vect\\_t sll](#) (const [vect\\_t](#) a)
- template<int s>  
static [INLINE CONST vect\\_t srl](#) (const [vect\\_t](#) a)
- template<uint32\_t s>  
static [INLINE CONST vect\\_t shuffle](#) (const [vect\\_t](#) a)
- static [INLINE CONST vect\\_t unpacklo\\_intrinsic](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpackhi\\_intrinsic](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpacklo](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpackhi](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE void unpacklohi](#) ([vect\\_t](#) &lo, [vect\\_t](#) &hi, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t pack\\_even](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t pack\\_odd](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE void pack](#) ([vect\\_t](#) &even, [vect\\_t](#) &odd, const [vect\\_t](#) a, const [vect\\_t](#) b)

- static `INLINE` void `transpose` (`vect_t` &r0, `vect_t` &r1, `vect_t` &r2, `vect_t` &r3, `vect_t` &r4, `vect_t` &r5, `vect_t` &r6, `vect_t` &r7)
- template<uint8\_t s>  
static `INLINE` `CONST` `vect_t` `blend` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `add` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `addin` (`vect_t` &a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `sub` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `subin` (`vect_t` &a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `mullo` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `mul` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `fmadd` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `fmaddin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `fnmadd` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `fnmaddin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `fmsub` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `fmsubin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `round` (const `vect_t` a)
- static `INLINE` `vect_t` `mod` (`vect_t` &C, const `vect_t` &P, const `__m64` &INVP, const `vect_t` &NEGP, const `vect_t` &MIN, const `vect_t` &MAX, `vect_t` &Q, `vect_t` &T)
- static `INLINE` `CONST` `vect_t` `zero` ()
- template<uint8\_t s>  
static `INLINE` `CONST` `vect_t` `sll128` (const `vect_t` a)
- template<uint8\_t s>  
static `INLINE` `CONST` `vect_t` `srl128` (const `vect_t` a)
- static `INLINE` `CONST` `vect_t` `vand` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `vor` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `vxor` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `vandnot` (const `vect_t` a, const `vect_t` b)

### Static Public Attributes

- static const constexpr size\_t `vect_size` = 8
- static const constexpr size\_t `alignment` = 16

## 12.277.1 Member Typedef Documentation

### 12.277.1.1 scalar\_t

```
using scalar_t = uint16_t
```

### 12.277.1.2 aligned\_allocator

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

### 12.277.1.3 aligned\_vector

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

### 12.277.1.4 is\_same\_element

```
template<class Field >  
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

### 12.277.1.5 vect\_t

```
using vect_t = __m128i [inherited]
```

## 12.277.2 Member Function Documentation

### 12.277.2.1 type\_string()

```
static const std::string type_string () [inline], [static]
```

### 12.277.2.2 set1() [1/2]

```
static INLINE CONST vect_t set1 (  
    const scalar_t x) [inline], [static]
```

### 12.277.2.3 set() [1/2]

```
static INLINE CONST vect_t set (  
    const scalar_t x0,  
    const scalar_t x1,  
    const scalar_t x2,  
    const scalar_t x3,  
    const scalar_t x4,  
    const scalar_t x5,  
    const scalar_t x6,  
    const scalar_t x7) [inline], [static]
```

### 12.277.2.4 gather() [1/2]

```
template<class T >  
static INLINE PURE vect_t gather (  
    const scalar_t *const p,  
    const T *const idx) [inline], [static]
```

### 12.277.2.5 load() [1/2]

```
static INLINE PURE vect_t load (  
    const scalar_t *const p) [inline], [static]
```

### 12.277.2.6 loadu() [1/2]

```
static INLINE PURE vect_t loadu (  
    const scalar_t *const p) [inline], [static]
```

### 12.277.2.7 store() [1/2]

```
static INLINE void store (  
    scalar_t * p,  
    vect_t v) [inline], [static]
```

### 12.277.2.8 storeu() [1/2]

```
static INLINE void storeu (  
    scalar_t * p,  
    vect_t v) [inline], [static]
```

### 12.277.2.9 stream() [1/2]

```
static INLINE void stream (  
    scalar_t * p,  
    const vect_t v) [inline], [static]
```

**12.277.2.10 sra()**

```
template<int s>
static INLINE CONST vect_t sra (
    const vect_t a) [inline], [static]
```

**12.277.2.11 greater()**

```
static INLINE CONST vect_t greater (
    vect_t a,
    vect_t b) [inline], [static]
```

**12.277.2.12 lesser()**

```
static INLINE CONST vect_t lesser (
    vect_t a,
    vect_t b) [inline], [static]
```

**12.277.2.13 greater\_eq()**

```
static INLINE CONST vect_t greater_eq (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.277.2.14 lesser\_eq()**

```
static INLINE CONST vect_t lesser_eq (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.277.2.15 mulhi()**

```
static INLINE CONST vect_t mulhi (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.277.2.16 mulx()**

```
static INLINE CONST vect_t mulx (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.277.2.17 fmaddx()**

```
static INLINE CONST vect_t fmaddx (
    const vect_t c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.277.2.18 fmaddxin()**

```
static INLINE vect_t fmaddxin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.277.2.19 fnmaddx()**

```
static INLINE CONST vect_t fnmaddx (
    const vect_t c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.277.2.20 fnmaddxin()**

```
static INLINE vect_t fnmaddxin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.277.2.21 fmsubx()**

```
static INLINE CONST vect_t fmsubx (
    const vect_t c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.277.2.22 fmsubxin()**

```
static INLINE vect_t fmsubxin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.277.2.23 hadd\_to\_scal()**

```
static INLINE CONST scalar_t hadd_to_scal (
    const vect_t a) [inline], [static]
```

**12.277.2.24 valid()**

```
template<class T >
static constexpr bool valid (
    T * p) [inline], [static], [constexpr], [inherited]
```

**12.277.2.25 compliant()**

```
template<class T >
static constexpr bool compliant (
    T n) [inline], [static], [constexpr], [inherited]
```

**12.277.2.26 set1() [2/2]**

```
static INLINE CONST vect_t set1 (
    const scalar_t x) [inline], [static], [inherited]
```

**12.277.2.27 set() [2/2]**

```
static INLINE CONST vect_t set (
    const scalar_t x0,
    const scalar_t x1,
    const scalar_t x2,
    const scalar_t x3,
    const scalar_t x4,
    const scalar_t x5,
```

```
    const scalar_t x6,  
    const scalar_t x7) [inline], [static], [inherited]
```

#### 12.277.2.28 gather() [2/2]

```
template<class T >  
static INLINE PURE vect_t gather (  
    const scalar_t *const p,  
    const T *const idx) [inline], [static], [inherited]
```

#### 12.277.2.29 load() [2/2]

```
static INLINE PURE vect_t load (  
    const scalar_t *const p) [inline], [static], [inherited]
```

#### 12.277.2.30 loadu() [2/2]

```
static INLINE PURE vect_t loadu (  
    const scalar_t *const p) [inline], [static], [inherited]
```

#### 12.277.2.31 store() [2/2]

```
static INLINE void store (  
    scalar_t * p,  
    vect_t v) [inline], [static], [inherited]
```

#### 12.277.2.32 storeu() [2/2]

```
static INLINE void storeu (  
    scalar_t * p,  
    vect_t v) [inline], [static], [inherited]
```

#### 12.277.2.33 stream() [2/2]

```
static INLINE void stream (  
    scalar_t * p,  
    const vect_t v) [inline], [static], [inherited]
```

#### 12.277.2.34 sll()

```
template<int s>  
static INLINE CONST vect_t sll (  
    const vect_t a) [inline], [static], [inherited]
```

#### 12.277.2.35 srl()

```
template<int s>  
static INLINE CONST vect_t srl (  
    const vect_t a) [inline], [static], [inherited]
```

#### 12.277.2.36 shuffle()

```
template<uint32_t s>  
static INLINE CONST vect_t shuffle (  
    const vect_t a) [inline], [static], [inherited]
```



**12.277.2.37 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.277.2.38 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.277.2.39 unpacklo()**

```
static INLINE CONST vect_t unpacklo (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.277.2.40 unpackhi()**

```
static INLINE CONST vect_t unpackhi (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.277.2.41 unpacklohi()**

```
static INLINE void unpacklohi (  
    vect_t & lo,  
    vect_t & hi,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.277.2.42 pack\_even()**

```
static INLINE CONST vect_t pack_even (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.277.2.43 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.277.2.44 pack()**

```
static INLINE void pack (  
    vect_t & even,  
    vect_t & odd,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.277.2.45 transpose()**

```
static INLINE void transpose (  
    vect_t & r0,  
    vect_t & r1,  
    vect_t & r2,  
    vect_t & r3,  
    vect_t & r4,
```

```
    vect_t & r5,  
    vect_t & r6,  
    vect_t & r7) [inline], [static], [inherited]
```

#### 12.277.2.46 blend()

```
template<uint8_t s>  
static INLINE CONST vect_t blend (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

#### 12.277.2.47 add()

```
static INLINE CONST vect_t add (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

#### 12.277.2.48 addin()

```
static INLINE vect_t addin (  
    vect_t & a,  
    const vect_t b) [inline], [static], [inherited]
```

#### 12.277.2.49 sub()

```
static INLINE CONST vect_t sub (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

#### 12.277.2.50 subin()

```
static INLINE vect_t subin (  
    vect_t & a,  
    const vect_t b) [inline], [static], [inherited]
```

#### 12.277.2.51 mullo()

```
static INLINE CONST vect_t mullo (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

#### 12.277.2.52 mul()

```
static INLINE CONST vect_t mul (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

#### 12.277.2.53 fmadd()

```
static INLINE CONST vect_t fmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

#### 12.277.2.54 fmaddin()

```
static INLINE vect_t fmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.277.2.55 fnmadd()**

```
static INLINE CONST vect_t fnmadd (
    const vect_t c,
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.277.2.56 fnmaddin()**

```
static INLINE vect_t fnmaddin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.277.2.57 fmsub()**

```
static INLINE CONST vect_t fmsub (
    const vect_t c,
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.277.2.58 fmsubin()**

```
static INLINE vect_t fmsubin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.277.2.59 eq()**

```
static INLINE CONST vect_t eq (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.277.2.60 round()**

```
static INLINE CONST vect_t round (
    const vect_t a) [inline], [static], [inherited]
```

**12.277.2.61 mod()**

```
static INLINE vect_t mod (
    vect_t & C,
    const vect_t & P,
    const __m64 & INVP,
    const vect_t & NEGP,
    const vect_t & MIN,
    const vect_t & MAX,
    vect_t & Q,
    vect_t & T) [inline], [static], [inherited]
```

**12.277.2.62 zero()**

```
static INLINE CONST vect_t zero () [inline], [static], [inherited]
```

**12.277.2.63 sll128()**

```
template<uint8_t s>
static INLINE CONST vect_t sll128 (
    const vect_t a) [inline], [static], [inherited]
```

**12.277.2.64 srl128()**

```
template<uint8_t s>
static INLINE CONST vect_t srl128 (
    const vect_t a) [inline], [static], [inherited]
```

**12.277.2.65 vand()**

```
static INLINE CONST vect_t vand (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.277.2.66 vor()**

```
static INLINE CONST vect_t vor (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.277.2.67 vxor()**

```
static INLINE CONST vect_t vxor (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.277.2.68 vandnot()**

```
static INLINE CONST vect_t vandnot (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.277.3 Field Documentation****12.277.3.1 vect\_size**

```
const constexpr size_t vect_size = 8 [static], [constexpr], [inherited]
```

**12.277.3.2 alignment**

```
const constexpr size_t alignment = 16 [static], [constexpr], [inherited]
```

The documentation for this struct was generated from the following file:

- [simd128\\_int16.inl](#)

**12.278 Simd128\_impl< true, true, false, 4 > Struct Reference**

Inheritance diagram for Simd128\_impl< true, true, false, 4 >:

**Data Structures**

- union [Converter](#)

## Public Types

- using `scalar_t` = `uint32_t`
- using `aligned_allocator` = `AlignedAllocator<scalar_t, Alignment(alignment)>`
- using `aligned_vector` = `std::vector<scalar_t, aligned_allocator>`
- template<class `Field` >  
using `is_same_element` = `std::is_same<typename Field::Element, scalar_t>`
- using `vect_t` = `__m128i`

## Static Public Member Functions

- static const std::string `type_string` ()
- static `INLINE CONST vect_t set1` (const `scalar_t` x)
- static `INLINE CONST vect_t set` (const `scalar_t` x0, const `scalar_t` x1, const `scalar_t` x2, const `scalar_t` x3)
- template<class `T` >  
static `INLINE PURE vect_t gather` (const `scalar_t` \*const p, const `T` \*const idx)
- static `INLINE PURE vect_t load` (const `scalar_t` \*const p)
- static `INLINE PURE vect_t loadu` (const `scalar_t` \*const p)
- static `INLINE void store` (`scalar_t` \*p, `vect_t` v)
- static `INLINE void storeu` (`scalar_t` \*p, `vect_t` v)
- static `INLINE void stream` (`scalar_t` \*p, const `vect_t` v)
- template<int `s`>  
static `INLINE CONST vect_t sra` (const `vect_t` a)
- static `INLINE CONST vect_t greater` (`vect_t` a, `vect_t` b)
- static `INLINE CONST vect_t lesser` (`vect_t` a, `vect_t` b)
- static `INLINE CONST vect_t greater_eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t lesser_eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t mulhi` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t mulx` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmaddx` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fmaddxin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fnmaddx` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fnmaddxin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmsubx` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fmsubxin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST scalar_t hadd_to_scal` (const `vect_t` a)
- template<class `T` >  
static constexpr bool `valid` (`T` \*p)
- template<class `T` >  
static constexpr bool `compliant` (`T` n)
- static `INLINE CONST vect_t set1` (const `scalar_t` x)
- static `INLINE CONST vect_t set` (const `scalar_t` x0, const `scalar_t` x1, const `scalar_t` x2, const `scalar_t` x3)
- template<class `T` >  
static `INLINE PURE vect_t gather` (const `scalar_t` \*const p, const `T` \*const idx)
- static `INLINE PURE vect_t load` (const `scalar_t` \*const p)
- static `INLINE PURE vect_t loadu` (const `scalar_t` \*const p)
- static `INLINE void store` (`scalar_t` \*p, `vect_t` v)
- static `INLINE void storeu` (`scalar_t` \*p, `vect_t` v)
- static `INLINE void stream` (`scalar_t` \*p, const `vect_t` v)
- template<int `s`>  
static `INLINE CONST vect_t sll` (const `vect_t` a)
- template<int `s`>  
static `INLINE CONST vect_t srl` (const `vect_t` a)
- template<uint8\_t `s`>  
static `INLINE CONST vect_t shuffle` (const `vect_t` a)
- static `INLINE CONST vect_t unpacklo_intrinsic` (const `vect_t` a, const `vect_t` b)

- static `INLINE CONST vect_t unpackhi_intrinsic` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t unpacklo` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t unpackhi` (const `vect_t` a, const `vect_t` b)
- static `INLINE void unpacklohi` (`vect_t` &lo, `vect_t` &hi, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t pack_even` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t pack_odd` (const `vect_t` a, const `vect_t` b)
- static `INLINE void pack` (`vect_t` &even, `vect_t` &odd, const `vect_t` a, const `vect_t` b)
- static `INLINE void transpose` (`vect_t` &r0, `vect_t` &r1, `vect_t` &r2, `vect_t` &r3)
- template<uint8\_t s>
  - static `INLINE CONST vect_t blend` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t add` (const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t addin` (`vect_t` &a, const `vect_t` b)
- static `INLINE CONST vect_t sub` (const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t subin` (`vect_t` &a, const `vect_t` b)
- static `INLINE CONST vect_t mullo` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t mul` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmadd` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fmaddin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fnmadd` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fnmaddin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmsub` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fmsubin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t round` (const `vect_t` a)
- static `INLINE vect_t mod` (`vect_t` &C, const `vect_t` &P, const `vect_t` &INVP, const `vect_t` &NEGP, const `vect_t` &MIN, const `vect_t` &MAX, `vect_t` &Q, `vect_t` &T)
- static `INLINE CONST vect_t zero` ()
- template<uint8\_t s>
  - static `INLINE CONST vect_t sll128` (const `vect_t` a)
- template<uint8\_t s>
  - static `INLINE CONST vect_t srl128` (const `vect_t` a)
- static `INLINE CONST vect_t vand` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t vor` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t vxor` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t vandnot` (const `vect_t` a, const `vect_t` b)

## Static Public Attributes

- static const constexpr size\_t `vect_size` = 4
- static const constexpr size\_t `alignment` = 16

## 12.278.1 Member Typedef Documentation

### 12.278.1.1 scalar\_t

```
using scalar_t = uint32_t
```

### 12.278.1.2 aligned\_allocator

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

### 12.278.1.3 aligned\_vector

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

**12.278.1.4 is\_same\_element**

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

**12.278.1.5 vect\_t**

```
using vect_t = __m128i [inherited]
```

**12.278.2 Member Function Documentation****12.278.2.1 type\_string()**

```
static const std::string type_string () [inline], [static]
```

**12.278.2.2 set1() [1/2]**

```
static INLINE CONST vect_t set1 (
    const scalar_t x) [inline], [static]
```

**12.278.2.3 set() [1/2]**

```
static INLINE CONST vect_t set (
    const scalar_t x0,
    const scalar_t x1,
    const scalar_t x2,
    const scalar_t x3) [inline], [static]
```

**12.278.2.4 gather() [1/2]**

```
template<class T >
static INLINE PURE vect_t gather (
    const scalar_t *const p,
    const T *const idx) [inline], [static]
```

**12.278.2.5 load() [1/2]**

```
static INLINE PURE vect_t load (
    const scalar_t *const p) [inline], [static]
```

**12.278.2.6 loadu() [1/2]**

```
static INLINE PURE vect_t loadu (
    const scalar_t *const p) [inline], [static]
```

**12.278.2.7 store() [1/2]**

```
static INLINE void store (
    scalar_t * p,
    vect_t v) [inline], [static]
```

**12.278.2.8 storeu() [1/2]**

```
static INLINE void storeu (
    scalar_t * p,
    vect_t v) [inline], [static]
```

**12.278.2.9 stream() [1/2]**

```
static INLINE void stream (  
    scalar_t * p,  
    const vect_t v) [inline], [static]
```

**12.278.2.10 sra()**

```
template<int s>  
static INLINE CONST vect_t sra (  
    const vect_t a) [inline], [static]
```

**12.278.2.11 greater()**

```
static INLINE CONST vect_t greater (  
    vect_t a,  
    vect_t b) [inline], [static]
```

**12.278.2.12 lesser()**

```
static INLINE CONST vect_t lesser (  
    vect_t a,  
    vect_t b) [inline], [static]
```

**12.278.2.13 greater\_eq()**

```
static INLINE CONST vect_t greater_eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.278.2.14 lesser\_eq()**

```
static INLINE CONST vect_t lesser_eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.278.2.15 mulhi()**

```
static INLINE CONST vect_t mulhi (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.278.2.16 mulx()**

```
static INLINE CONST vect_t mulx (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.278.2.17 fmaddx()**

```
static INLINE CONST vect_t fmaddx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.278.2.18 fmaddxin()**

```
static INLINE vect_t fmaddxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```



**12.278.2.19 fnmaddx()**

```
static INLINE CONST vect_t fnmaddx (
    const vect_t c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.278.2.20 fnmaddxin()**

```
static INLINE vect_t fnmaddxin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.278.2.21 fmsubx()**

```
static INLINE CONST vect_t fmsubx (
    const vect_t c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.278.2.22 fmsubxin()**

```
static INLINE vect_t fmsubxin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.278.2.23 hadd\_to\_scal()**

```
static INLINE CONST scalar_t hadd_to_scal (
    const vect_t a) [inline], [static]
```

**12.278.2.24 valid()**

```
template<class T >
static constexpr bool valid (
    T * p) [inline], [static], [constexpr], [inherited]
```

**12.278.2.25 compliant()**

```
template<class T >
static constexpr bool compliant (
    T n) [inline], [static], [constexpr], [inherited]
```

**12.278.2.26 set1() [2/2]**

```
static INLINE CONST vect_t set1 (
    const scalar_t x) [inline], [static], [inherited]
```

**12.278.2.27 set() [2/2]**

```
static INLINE CONST vect_t set (
    const scalar_t x0,
    const scalar_t x1,
    const scalar_t x2,
    const scalar_t x3) [inline], [static], [inherited]
```

**12.278.2.28 gather()** [2/2]

```
template<class T >
static INLINE PURE vect_t gather (
    const scalar_t *const p,
    const T *const idx) [inline], [static], [inherited]
```

**12.278.2.29 load()** [2/2]

```
static INLINE PURE vect_t load (
    const scalar_t *const p) [inline], [static], [inherited]
```

**12.278.2.30 loadu()** [2/2]

```
static INLINE PURE vect_t loadu (
    const scalar_t *const p) [inline], [static], [inherited]
```

**12.278.2.31 store()** [2/2]

```
static INLINE void store (
    scalar_t * p,
    vect_t v) [inline], [static], [inherited]
```

**12.278.2.32 storeu()** [2/2]

```
static INLINE void storeu (
    scalar_t * p,
    vect_t v) [inline], [static], [inherited]
```

**12.278.2.33 stream()** [2/2]

```
static INLINE void stream (
    scalar_t * p,
    const vect_t v) [inline], [static], [inherited]
```

**12.278.2.34 sll()**

```
template<int s>
static INLINE CONST vect_t sll (
    const vect_t a) [inline], [static], [inherited]
```

**12.278.2.35 srl()**

```
template<int s>
static INLINE CONST vect_t srl (
    const vect_t a) [inline], [static], [inherited]
```

**12.278.2.36 shuffle()**

```
template<uint8_t s>
static INLINE CONST vect_t shuffle (
    const vect_t a) [inline], [static], [inherited]
```

**12.278.2.37 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.38 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.39 unpacklo()**

```
static INLINE CONST vect_t unpacklo (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.40 unpackhi()**

```
static INLINE CONST vect_t unpackhi (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.41 unpacklohi()**

```
static INLINE void unpacklohi (  
    vect_t & lo,  
    vect_t & hi,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.42 pack\_even()**

```
static INLINE CONST vect_t pack_even (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.43 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.44 pack()**

```
static INLINE void pack (  
    vect_t & even,  
    vect_t & odd,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.45 transpose()**

```
static INLINE void transpose (  
    vect_t & r0,  
    vect_t & r1,  
    vect_t & r2,  
    vect_t & r3) [inline], [static], [inherited]
```

**12.278.2.46 blend()**

```
template<uint8_t s>  
static INLINE CONST vect_t blend (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.47 add()**

```
static INLINE CONST vect_t add (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.48 addin()**

```
static INLINE vect_t addin (  
    vect_t & a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.49 sub()**

```
static INLINE CONST vect_t sub (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.50 subin()**

```
static INLINE vect_t subin (  
    vect_t & a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.51 mullo()**

```
static INLINE CONST vect_t mullo (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.52 mul()**

```
static INLINE CONST vect_t mul (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.53 fmadd()**

```
static INLINE CONST vect_t fmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.54 fmaddin()**

```
static INLINE vect_t fmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.55 fnmadd()**

```
static INLINE CONST vect_t fnmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.56 fnmaddin()**

```
static INLINE vect_t fnmaddin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.57 fmsub()**

```
static INLINE CONST vect_t fmsub (
    const vect_t c,
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.58 fmsubin()**

```
static INLINE vect_t fmsubin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.59 eq()**

```
static INLINE CONST vect_t eq (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.60 round()**

```
static INLINE CONST vect_t round (
    const vect_t a) [inline], [static], [inherited]
```

**12.278.2.61 mod()**

```
static INLINE vect_t mod (
    vect_t & C,
    const vect_t & P,
    const vect_t & INVP,
    const vect_t & NEGP,
    const vect_t & MIN,
    const vect_t & MAX,
    vect_t & Q,
    vect_t & T) [inline], [static], [inherited]
```

**12.278.2.62 zero()**

```
static INLINE CONST vect_t zero () [inline], [static], [inherited]
```

**12.278.2.63 sll128()**

```
template<uint8_t s>
static INLINE CONST vect_t sll128 (
    const vect_t a) [inline], [static], [inherited]
```

**12.278.2.64 srl128()**

```
template<uint8_t s>
static INLINE CONST vect_t srl128 (
    const vect_t a) [inline], [static], [inherited]
```

**12.278.2.65 vand()**

```
static INLINE CONST vect_t vand (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.66 vor()**

```
static INLINE CONST vect_t vor (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.67 vxor()**

```
static INLINE CONST vect_t vxor (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.278.2.68 vandnot()**

```
static INLINE CONST vect_t vandnot (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.278.3 Field Documentation****12.278.3.1 vect\_size**

```
const constexpr size_t vect_size = 4 [static], [constexpr], [inherited]
```

**12.278.3.2 alignment**

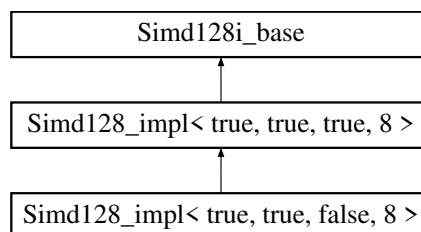
```
const constexpr size_t alignment = 16 [static], [constexpr], [inherited]
```

The documentation for this struct was generated from the following file:

- [simd128\\_int32.inl](#)

**12.279 Simd128\_impl< true, true, false, 8 > Struct Reference**

Inheritance diagram for Simd128\_impl< true, true, false, 8 >:

**Data Structures**

- union [Converter](#)

**Public Types**

- using [scalar\\_t](#) = uint64\_t
- using [aligned\\_allocator](#) = AlignedAllocator<[scalar\\_t](#), Alignment([alignment](#))>
- using [aligned\\_vector](#) = std::vector<[scalar\\_t](#), [aligned\\_allocator](#)>

- template<class Field >  
using is\_same\_element = std::is\_same<typename Field::Element, scalar\_t>
- using vect\_t = \_\_m128i

### Static Public Member Functions

- static const std::string type\_string ()
- static INLINE CONST vect\_t set1 (const scalar\_t x)
- static INLINE CONST vect\_t set (const scalar\_t x0, const scalar\_t x1)
- template<class T >  
static INLINE PURE vect\_t gather (const scalar\_t \*const p, const T \*const idx)
- static INLINE PURE vect\_t load (const scalar\_t \*const p)
- static INLINE PURE vect\_t loadu (const scalar\_t \*const p)
- static INLINE void store (scalar\_t \*p, vect\_t v)
- static INLINE void storeu (scalar\_t \*p, vect\_t v)
- static INLINE void stream (scalar\_t \*p, const vect\_t v)
- template<int s>  
static INLINE CONST vect\_t sra (const vect\_t a)
- static INLINE CONST vect\_t greater (vect\_t a, vect\_t b)
- static INLINE CONST vect\_t lesser (vect\_t a, vect\_t b)
- static INLINE CONST vect\_t greater\_eq (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t lesser\_eq (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t mullo (const vect\_t x0, const vect\_t x1)
- static INLINE CONST vect\_t mulhi (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t mulx (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmaddx (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmaddxin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fnmaddx (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fnmaddxin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmsubx (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmsubxin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST scalar\_t hadd\_to\_scal (const vect\_t a)
- template<class T >  
static constexpr bool valid (T \*p)
- template<class T >  
static constexpr bool compliant (T n)
- static INLINE CONST vect\_t set1 (const scalar\_t x)
- static INLINE CONST vect\_t set (const scalar\_t x0, const scalar\_t x1)
- template<class T >  
static INLINE PURE vect\_t gather (const scalar\_t \*const p, const T \*const idx)
- template<int idx>  
static INLINE CONST scalar\_t get (vect\_t v)
- static INLINE PURE vect\_t load (const scalar\_t \*const p)
- static INLINE PURE vect\_t loadu (const scalar\_t \*const p)
- static INLINE void store (scalar\_t \*p, vect\_t v)
- static INLINE void storeu (scalar\_t \*p, vect\_t v)
- static INLINE void stream (scalar\_t \*p, const vect\_t v)
- template<int s>  
static INLINE CONST vect\_t sll (const vect\_t a)
- template<int s>  
static INLINE CONST vect\_t srl (const vect\_t a)
- template<uint8\_t s>  
static INLINE CONST vect\_t shuffle (const vect\_t a)
- static INLINE CONST vect\_t unpacklo\_intrinsic (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpackhi\_intrinsic (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpacklo (const vect\_t a, const vect\_t b)

- static `INLINE CONST vect_t unpackhi` (const `vect_t` a, const `vect_t` b)
- static `INLINE` void `unpacklohi` (`vect_t` &lo, `vect_t` &hi, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t pack_even` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t pack_odd` (const `vect_t` a, const `vect_t` b)
- static `INLINE` void `pack` (`vect_t` &even, `vect_t` &odd, const `vect_t` a, const `vect_t` b)
- static `INLINE` void `transpose` (`vect_t` &r0, `vect_t` &r1)
- template<uint8\_t s>
  - static `INLINE CONST vect_t blend` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t add` (const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t addin` (`vect_t` &a, const `vect_t` b)
- static `INLINE CONST vect_t sub` (const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t subin` (`vect_t` &a, const `vect_t` b)
- static `INLINE CONST vect_t mul` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmadd` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fmaddin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fnmadd` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fnmaddin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmsub` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fmsubin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t round` (const `vect_t` a)
- static `INLINE CONST vect_t mask_high` ()
- static `INLINE CONST vect_t mulhi_fast` (`vect_t` x, `vect_t` y)
- static `INLINE vect_t mod` (`vect_t` &C, const \_\_m128d &P, const \_\_m128d &INVP, const \_\_m128d &NEGP, const `vect_t` &POW50REM, const \_\_m128d &MIN, const \_\_m128d &MAX, \_\_m128d &Q, \_\_m128d &T)
- static `INLINE CONST vect_t zero` ()
- template<uint8\_t s>
  - static `INLINE CONST vect_t sll128` (const `vect_t` a)
- template<uint8\_t s>
  - static `INLINE CONST vect_t srl128` (const `vect_t` a)
- static `INLINE CONST vect_t vand` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t vor` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t vxor` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t vandnot` (const `vect_t` a, const `vect_t` b)

### Static Public Attributes

- static const constexpr size\_t `vect_size` = 2
- static const constexpr size\_t `alignment` = 16

### Static Protected Member Functions

- static `INLINE CONST vect_t signbits` (const `vect_t` x)

## 12.279.1 Member Typedef Documentation

### 12.279.1.1 scalar\_t

```
using scalar_t = uint64_t
```

### 12.279.1.2 aligned\_allocator

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

### 12.279.1.3 aligned\_vector

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```



#### 12.279.1.4 is\_same\_element

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

#### 12.279.1.5 vect\_t

```
using vect_t = __m128i [inherited]
```

### 12.279.2 Member Function Documentation

#### 12.279.2.1 type\_string()

```
static const std::string type_string () [inline], [static]
```

#### 12.279.2.2 set1() [1/2]

```
static INLINE CONST vect_t set1 (
    const scalar_t x) [inline], [static]
```

#### 12.279.2.3 set() [1/2]

```
static INLINE CONST vect_t set (
    const scalar_t x0,
    const scalar_t x1) [inline], [static]
```

#### 12.279.2.4 gather() [1/2]

```
template<class T >
static INLINE PURE vect_t gather (
    const scalar_t *const p,
    const T *const idx) [inline], [static]
```

#### 12.279.2.5 load() [1/2]

```
static INLINE PURE vect_t load (
    const scalar_t *const p) [inline], [static]
```

#### 12.279.2.6 loadu() [1/2]

```
static INLINE PURE vect_t loadu (
    const scalar_t *const p) [inline], [static]
```

#### 12.279.2.7 store() [1/2]

```
static INLINE void store (
    scalar_t * p,
    vect_t v) [inline], [static]
```

#### 12.279.2.8 storeu() [1/2]

```
static INLINE void storeu (
    scalar_t * p,
    vect_t v) [inline], [static]
```

#### 12.279.2.9 stream() [1/2]

```
static INLINE void stream (
    scalar_t * p,
    const vect_t v) [inline], [static]
```

**12.279.2.10 sra()**

```
template<int s>
static INLINE CONST vect_t sra (
    const vect_t a) [inline], [static]
```

**12.279.2.11 greater()**

```
static INLINE CONST vect_t greater (
    vect_t a,
    vect_t b) [inline], [static]
```

**12.279.2.12 lesser()**

```
static INLINE CONST vect_t lesser (
    vect_t a,
    vect_t b) [inline], [static]
```

**12.279.2.13 greater\_eq()**

```
static INLINE CONST vect_t greater_eq (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.279.2.14 lesser\_eq()**

```
static INLINE CONST vect_t lesser_eq (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.279.2.15 mullo()**

```
static INLINE CONST vect_t mullo (
    const vect_t x0,
    const vect_t x1) [inline], [static]
```

**12.279.2.16 mulhi()**

```
static INLINE CONST vect_t mulhi (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.279.2.17 mulx()**

```
static INLINE CONST vect_t mulx (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.279.2.18 fmaddx()**

```
static INLINE CONST vect_t fmaddx (
    const vect_t c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.279.2.19 fmaddxin()**

```
static INLINE vect_t fmaddxin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.279.2.20 fnmaddx()**

```
static INLINE CONST vect_t fnmaddx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.279.2.21 fnmaddxin()**

```
static INLINE vect_t fnmaddxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.279.2.22 fmsubx()**

```
static INLINE CONST vect_t fmsubx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.279.2.23 fmsubxin()**

```
static INLINE vect_t fmsubxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.279.2.24 hadd\_to\_scal()**

```
static INLINE CONST scalar_t hadd_to_scal (  
    const vect_t a) [inline], [static]
```

**12.279.2.25 valid()**

```
template<class T >  
static constexpr bool valid (  
    T * p) [inline], [static], [constexpr], [inherited]
```

**12.279.2.26 compliant()**

```
template<class T >  
static constexpr bool compliant (  
    T n) [inline], [static], [constexpr], [inherited]
```

**12.279.2.27 set1() [2/2]**

```
static INLINE CONST vect_t set1 (  
    const scalar_t x) [inline], [static], [inherited]
```

**12.279.2.28 set() [2/2]**

```
static INLINE CONST vect_t set (  
    const scalar_t x0,  
    const scalar_t x1) [inline], [static], [inherited]
```

**12.279.2.29 gather()** [2/2]

```
template<class T >
static INLINE PURE vect_t gather (
    const scalar_t *const p,
    const T *const idx) [inline], [static], [inherited]
```

**12.279.2.30 get()**

```
template<int idx>
static INLINE CONST scalar_t get (
    vect_t v) [inline], [static], [inherited]
```

**12.279.2.31 load()** [2/2]

```
static INLINE PURE vect_t load (
    const scalar_t *const p) [inline], [static], [inherited]
```

**12.279.2.32 loadu()** [2/2]

```
static INLINE PURE vect_t loadu (
    const scalar_t *const p) [inline], [static], [inherited]
```

**12.279.2.33 store()** [2/2]

```
static INLINE void store (
    scalar_t * p,
    vect_t v) [inline], [static], [inherited]
```

**12.279.2.34 storeu()** [2/2]

```
static INLINE void storeu (
    scalar_t * p,
    vect_t v) [inline], [static], [inherited]
```

**12.279.2.35 stream()** [2/2]

```
static INLINE void stream (
    scalar_t * p,
    const vect_t v) [inline], [static], [inherited]
```

**12.279.2.36 sll()**

```
template<int s>
static INLINE CONST vect_t sll (
    const vect_t a) [inline], [static], [inherited]
```

**12.279.2.37 srl()**

```
template<int s>
static INLINE CONST vect_t srl (
    const vect_t a) [inline], [static], [inherited]
```

**12.279.2.38 shuffle()**

```
template<uint8_t s>
static INLINE CONST vect_t shuffle (
    const vect_t a) [inline], [static], [inherited]
```

**12.279.2.39 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.279.2.40 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.279.2.41 unpacklo()**

```
static INLINE CONST vect_t unpacklo (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.279.2.42 unpackhi()**

```
static INLINE CONST vect_t unpackhi (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.279.2.43 unpacklohi()**

```
static INLINE void unpacklohi (  
    vect_t & lo,  
    vect_t & hi,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.279.2.44 pack\_even()**

```
static INLINE CONST vect_t pack_even (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.279.2.45 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.279.2.46 pack()**

```
static INLINE void pack (  
    vect_t & even,  
    vect_t & odd,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.279.2.47 transpose()**

```
static INLINE void transpose (  
    vect_t & r0,  
    vect_t & r1) [inline], [static], [inherited]
```

**12.279.2.48 blend()**

```
template<uint8_t s>
static INLINE CONST vect_t blend (
    const vect_t a,
    const vect_t b)  [inline], [static], [inherited]
```

**12.279.2.49 add()**

```
static INLINE CONST vect_t add (
    const vect_t a,
    const vect_t b)  [inline], [static], [inherited]
```

**12.279.2.50 addin()**

```
static INLINE vect_t addin (
    vect_t & a,
    const vect_t b)  [inline], [static], [inherited]
```

**12.279.2.51 sub()**

```
static INLINE CONST vect_t sub (
    const vect_t a,
    const vect_t b)  [inline], [static], [inherited]
```

**12.279.2.52 subin()**

```
static INLINE vect_t subin (
    vect_t & a,
    const vect_t b)  [inline], [static], [inherited]
```

**12.279.2.53 mul()**

```
static INLINE CONST vect_t mul (
    const vect_t a,
    const vect_t b)  [inline], [static], [inherited]
```

**12.279.2.54 fmadd()**

```
static INLINE CONST vect_t fmadd (
    const vect_t c,
    const vect_t a,
    const vect_t b)  [inline], [static], [inherited]
```

**12.279.2.55 fmaddin()**

```
static INLINE vect_t fmaddin (
    vect_t & c,
    const vect_t a,
    const vect_t b)  [inline], [static], [inherited]
```

**12.279.2.56 fnmadd()**

```
static INLINE CONST vect_t fnmadd (
    const vect_t c,
    const vect_t a,
    const vect_t b)  [inline], [static], [inherited]
```

**12.279.2.57 fnmaddin()**

```
static INLINE vect_t fnmaddin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.279.2.58 fmsub()**

```
static INLINE CONST vect_t fmsub (
    const vect_t c,
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.279.2.59 fmsubin()**

```
static INLINE vect_t fmsubin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.279.2.60 eq()**

```
static INLINE CONST vect_t eq (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.279.2.61 round()**

```
static INLINE CONST vect_t round (
    const vect_t a) [inline], [static], [inherited]
```

**12.279.2.62 mask\_high()**

```
static INLINE CONST vect_t mask_high () [inline], [static], [inherited]
```

**12.279.2.63 mulhi\_fast()**

```
INLINE CONST vect_t mulhi_fast (
    vect_t x,
    vect_t y) [static], [inherited]
```

**12.279.2.64 mod()**

```
INLINE vect_t mod (
    vect_t & C,
    const __m128d & P,
    const __m128d & INVP,
    const __m128d & NEGP,
    const vect_t & POW50REM,
    const __m128d & MIN,
    const __m128d & MAX,
    __m128d & Q,
    __m128d & T) [static], [inherited]
```

**12.279.2.65 signbits()**

```
static INLINE CONST vect_t signbits (
    const vect_t x) [inline], [static], [protected], [inherited]
```

**12.279.2.66 zero()**

```
static INLINE CONST vect_t zero () [inline], [static], [inherited]
```

**12.279.2.67 sll128()**

```
template<uint8_t s>
static INLINE CONST vect_t sll128 (
    const vect_t a) [inline], [static], [inherited]
```

**12.279.2.68 srl128()**

```
template<uint8_t s>
static INLINE CONST vect_t srl128 (
    const vect_t a) [inline], [static], [inherited]
```

**12.279.2.69 vand()**

```
static INLINE CONST vect_t vand (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.279.2.70 vor()**

```
static INLINE CONST vect_t vor (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.279.2.71 vxor()**

```
static INLINE CONST vect_t vxor (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.279.2.72 vandnot()**

```
static INLINE CONST vect_t vandnot (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.279.3 Field Documentation****12.279.3.1 vect\_size**

```
const constexpr size_t vect_size = 2 [static], [constexpr], [inherited]
```

**12.279.3.2 alignment**

```
const constexpr size_t alignment = 16 [static], [constexpr], [inherited]
```

The documentation for this struct was generated from the following file:

- [simd128\\_int64.inl](#)

**12.280 Simd128\_impl< true, true, true, 2 > Struct Reference**

Inheritance diagram for Simd128\_impl< true, true, true, 2 >:





## Data Structures

- union [Converter](#)

## Public Types

- using [vect\\_t](#) = \_\_m128i
- using [scalar\\_t](#) = int16\_t
- using [aligned\\_allocator](#) = AlignedAllocator<[scalar\\_t](#), Alignment([alignment](#))>
- using [aligned\\_vector](#) = std::vector<[scalar\\_t](#), [aligned\\_allocator](#)>
- template<class [Field](#) >  
using [is\\_same\\_element](#) = std::is\_same<typename [Field::Element](#), [scalar\\_t](#)>

## Static Public Member Functions

- static const std::string [type\\_string](#) ()
- template<class T >  
static constexpr bool [valid](#) (T \*p)
- template<class T >  
static constexpr bool [compliant](#) (T n)
- static [INLINE CONST vect\\_t set1](#) (const [scalar\\_t](#) x)
- static [INLINE CONST vect\\_t set](#) (const [scalar\\_t](#) x0, const [scalar\\_t](#) x1, const [scalar\\_t](#) x2, const [scalar\\_t](#) x3, const [scalar\\_t](#) x4, const [scalar\\_t](#) x5, const [scalar\\_t](#) x6, const [scalar\\_t](#) x7)
- template<class T >  
static [INLINE PURE vect\\_t gather](#) (const [scalar\\_t](#) \*const p, const T \*const idx)
- static [INLINE PURE vect\\_t load](#) (const [scalar\\_t](#) \*const p)
- static [INLINE PURE vect\\_t loadu](#) (const [scalar\\_t](#) \*const p)
- static [INLINE void store](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)
- static [INLINE void storeu](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)
- static [INLINE void stream](#) ([scalar\\_t](#) \*p, const [vect\\_t](#) v)
- template<int s>  
static [INLINE CONST vect\\_t sll](#) (const [vect\\_t](#) a)
- template<int s>  
static [INLINE CONST vect\\_t srl](#) (const [vect\\_t](#) a)
- template<int s>  
static [INLINE CONST vect\\_t sra](#) (const [vect\\_t](#) a)
- template<uint32\_t s>  
static [INLINE CONST vect\\_t shuffle](#) (const [vect\\_t](#) a)
- static [INLINE CONST vect\\_t unpacklo\\_intrinsic](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpackhi\\_intrinsic](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpacklo](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpackhi](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE void unpacklohi](#) ([vect\\_t](#) &lo, [vect\\_t](#) &hi, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t pack\\_even](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t pack\\_odd](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE void pack](#) ([vect\\_t](#) &even, [vect\\_t](#) &odd, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE void transpose](#) ([vect\\_t](#) &r0, [vect\\_t](#) &r1, [vect\\_t](#) &r2, [vect\\_t](#) &r3, [vect\\_t](#) &r4, [vect\\_t](#) &r5, [vect\\_t](#) &r6, [vect\\_t](#) &r7)

- `template<uint8_t s>`  
static `INLINE CONST vect_t blend` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t add` (const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t addin` (`vect_t` &a, const `vect_t` b)
- static `INLINE CONST vect_t sub` (const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t subin` (`vect_t` &a, const `vect_t` b)
- static `INLINE CONST vect_t mullo` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t mul` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t mulhi` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t mulx` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmaddd` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fmadddin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmadddx` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fmadddxin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fnmadd` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fnmaddin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fnmaddx` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fnmaddxin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmsub` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fmsubin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmsubx` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fmsubxin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t greater` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t lesser` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t greater_eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t lesser_eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST scalar_t hadd_to_scal` (const `vect_t` a)
- static `INLINE CONST vect_t round` (const `vect_t` a)
- static `INLINE vect_t mod` (`vect_t` &C, const `vect_t` &P, const `__m64` &INVP, const `vect_t` &NEGP, const `vect_t` &MIN, const `vect_t` &MAX, `vect_t` &Q, `vect_t` &T)
- static `INLINE CONST vect_t zero` ()
- `template<uint8_t s>`  
static `INLINE CONST vect_t sll128` (const `vect_t` a)
- `template<uint8_t s>`  
static `INLINE CONST vect_t srl128` (const `vect_t` a)
- static `INLINE CONST vect_t vand` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t vor` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t vxor` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t vandnot` (const `vect_t` a, const `vect_t` b)

### Static Public Attributes

- static const constexpr size\_t `vect_size` = 8
- static const constexpr size\_t `alignment` = 16

## 12.280.1 Member Typedef Documentation

### 12.280.1.1 vect\_t

using `vect_t` = `__m128i`

### 12.280.1.2 scalar\_t

using `scalar_t` = `int16_t`

**12.280.1.3 aligned\_allocator**

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

**12.280.1.4 aligned\_vector**

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

**12.280.1.5 is\_same\_element**

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

**12.280.2 Member Function Documentation****12.280.2.1 type\_string()**

```
static const std::string type_string () [inline], [static]
```

**12.280.2.2 valid()**

```
template<class T >
static constexpr bool valid (
    T * p) [inline], [static], [constexpr]
```

**12.280.2.3 compliant()**

```
template<class T >
static constexpr bool compliant (
    T n) [inline], [static], [constexpr]
```

**12.280.2.4 set1()**

```
static INLINE CONST vect_t set1 (
    const scalar_t x) [inline], [static]
```

**12.280.2.5 set()**

```
static INLINE CONST vect_t set (
    const scalar_t x0,
    const scalar_t x1,
    const scalar_t x2,
    const scalar_t x3,
    const scalar_t x4,
    const scalar_t x5,
    const scalar_t x6,
    const scalar_t x7) [inline], [static]
```

**12.280.2.6 gather()**

```
template<class T >
static INLINE PURE vect_t gather (
    const scalar_t *const p,
    const T *const idx) [inline], [static]
```

**12.280.2.7 load()**

```
static INLINE PURE vect_t load (
    const scalar_t *const p) [inline], [static]
```

**12.280.2.8 loadu()**

```
static INLINE PURE vect_t loadu (
    const scalar_t *const p) [inline], [static]
```

**12.280.2.9 store()**

```
static INLINE void store (
    scalar_t * p,
    vect_t v) [inline], [static]
```

**12.280.2.10 storeu()**

```
static INLINE void storeu (
    scalar_t * p,
    vect_t v) [inline], [static]
```

**12.280.2.11 stream()**

```
static INLINE void stream (
    scalar_t * p,
    const vect_t v) [inline], [static]
```

**12.280.2.12 sll()**

```
template<int s>
static INLINE CONST vect_t sll (
    const vect_t a) [inline], [static]
```

**12.280.2.13 srl()**

```
template<int s>
static INLINE CONST vect_t srl (
    const vect_t a) [inline], [static]
```

**12.280.2.14 sra()**

```
template<int s>
static INLINE CONST vect_t sra (
    const vect_t a) [inline], [static]
```

**12.280.2.15 shuffle()**

```
template<uint32_t s>
static INLINE CONST vect_t shuffle (
    const vect_t a) [inline], [static]
```

**12.280.2.16 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.280.2.17 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.280.2.18 unpacklo()**

```
static INLINE CONST vect_t unpacklo (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.280.2.19 unpackhi()**

```
static INLINE CONST vect_t unpackhi (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.280.2.20 unpacklohi()**

```
static INLINE void unpacklohi (  
    vect_t & lo,  
    vect_t & hi,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.280.2.21 pack\_even()**

```
static INLINE CONST vect_t pack_even (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.280.2.22 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.280.2.23 pack()**

```
static INLINE void pack (  
    vect_t & even,  
    vect_t & odd,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.280.2.24 transpose()**

```
static INLINE void transpose (  
    vect_t & r0,  
    vect_t & r1,  
    vect_t & r2,  
    vect_t & r3,  
    vect_t & r4,  
    vect_t & r5,  
    vect_t & r6,  
    vect_t & r7) [inline], [static]
```

**12.280.2.25 blend()**

```
template<uint8_t s>  
static INLINE CONST vect_t blend (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.280.2.26 add()**

```
static INLINE CONST vect_t add (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.280.2.27 addin()**

```
static INLINE vect_t addin (  
    vect_t & a,  
    const vect_t b) [inline], [static]
```

**12.280.2.28 sub()**

```
static INLINE CONST vect_t sub (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.280.2.29 subin()**

```
static INLINE vect_t subin (  
    vect_t & a,  
    const vect_t b) [inline], [static]
```

**12.280.2.30 mullo()**

```
static INLINE CONST vect_t mullo (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.280.2.31 mul()**

```
static INLINE CONST vect_t mul (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.280.2.32 mulhi()**

```
static INLINE CONST vect_t mulhi (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.280.2.33 mulx()**

```
static INLINE CONST vect_t mulx (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.280.2.34 fmadd()**

```
static INLINE CONST vect_t fmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.280.2.35 fmaddin()**

```
static INLINE vect_t fmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.280.2.36 fmaddx()**

```
static INLINE CONST vect_t fmaddx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.280.2.37 fmaddxin()**

```
static INLINE vect_t fmaddxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.280.2.38 fnmadd()**

```
static INLINE CONST vect_t fnmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.280.2.39 fnmaddin()**

```
static INLINE vect_t fnmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.280.2.40 fnmaddx()**

```
static INLINE CONST vect_t fnmaddx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.280.2.41 fnmaddxin()**

```
static INLINE vect_t fnmaddxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.280.2.42 fmsub()**

```
static INLINE CONST vect_t fmsub (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.280.2.43 fmsubin()**

```
static INLINE vect_t fmsubin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.280.2.44 fmsubx()**

```
static INLINE CONST vect_t fmsubx (  
    const vect_t c,
```

```

    const vect_t a,
    const vect_t b) [inline], [static]

```

#### 12.280.2.45 fmsubxin()

```

static INLINE vect_t fmsubxin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static]

```

#### 12.280.2.46 eq()

```

static INLINE CONST vect_t eq (
    const vect_t a,
    const vect_t b) [inline], [static]

```

#### 12.280.2.47 greater()

```

static INLINE CONST vect_t greater (
    const vect_t a,
    const vect_t b) [inline], [static]

```

#### 12.280.2.48 lesser()

```

static INLINE CONST vect_t lesser (
    const vect_t a,
    const vect_t b) [inline], [static]

```

#### 12.280.2.49 greater\_eq()

```

static INLINE CONST vect_t greater_eq (
    const vect_t a,
    const vect_t b) [inline], [static]

```

#### 12.280.2.50 lesser\_eq()

```

static INLINE CONST vect_t lesser_eq (
    const vect_t a,
    const vect_t b) [inline], [static]

```

#### 12.280.2.51 hadd\_to\_scal()

```

static INLINE CONST scalar_t hadd_to_scal (
    const vect_t a) [inline], [static]

```

#### 12.280.2.52 round()

```

static INLINE CONST vect_t round (
    const vect_t a) [inline], [static]

```

#### 12.280.2.53 mod()

```

static INLINE vect_t mod (
    vect_t & C,
    const vect_t & P,
    const __m64 & INV_P,
    const vect_t & NEG_P,
    const vect_t & MIN,
    const vect_t & MAX,

```



```
vect_t & Q,  
vect_t & T) [inline], [static]
```

#### 12.280.2.54 zero()

```
static INLINE CONST vect_t zero () [inline], [static], [inherited]
```

#### 12.280.2.55 sll128()

```
template<uint8_t s>  
static INLINE CONST vect_t sll128 (  
    const vect_t a) [inline], [static], [inherited]
```

#### 12.280.2.56 srl128()

```
template<uint8_t s>  
static INLINE CONST vect_t srl128 (  
    const vect_t a) [inline], [static], [inherited]
```

#### 12.280.2.57 vand()

```
static INLINE CONST vect_t vand (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

#### 12.280.2.58 vor()

```
static INLINE CONST vect_t vor (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

#### 12.280.2.59 vxor()

```
static INLINE CONST vect_t vxor (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

#### 12.280.2.60 vandnot()

```
static INLINE CONST vect_t vandnot (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

### 12.280.3 Field Documentation

#### 12.280.3.1 vect\_size

```
const constexpr size_t vect_size = 8 [static], [constexpr]
```

#### 12.280.3.2 alignment

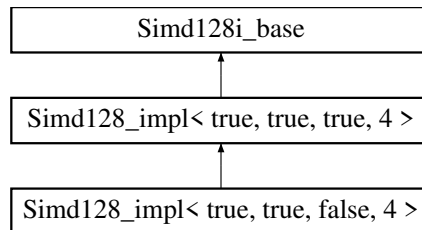
```
const constexpr size_t alignment = 16 [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [simd128\\_int16.inl](#)

## 12.281 Simd128\_impl< true, true, true, 4 > Struct Reference

Inheritance diagram for Simd128\_impl< true, true, true, 4 >:



### Data Structures

- union [Converter](#)

### Public Types

- using [vect\\_t](#) = \_\_m128i
- using [scalar\\_t](#) = int32\_t
- using [aligned\\_allocator](#) = AlignedAllocator<[scalar\\_t](#), Alignment([alignment](#))>
- using [aligned\\_vector](#) = std::vector<[scalar\\_t](#), [aligned\\_allocator](#)>
- template<class [Field](#) >  
using [is\\_same\\_element](#) = std::is\_same<typename [Field::Element](#), [scalar\\_t](#)>

### Static Public Member Functions

- static const std::string [type\\_string](#) ()
- template<class T >  
static constexpr bool [valid](#) (T \*p)
- template<class T >  
static constexpr bool [compliant](#) (T n)
- static [INLINE CONST vect\\_t set1](#) (const [scalar\\_t](#) x)
- static [INLINE CONST vect\\_t set](#) (const [scalar\\_t](#) x0, const [scalar\\_t](#) x1, const [scalar\\_t](#) x2, const [scalar\\_t](#) x3)
- template<class T >  
static [INLINE PURE vect\\_t gather](#) (const [scalar\\_t](#) \*const p, const T \*const idx)
- static [INLINE PURE vect\\_t load](#) (const [scalar\\_t](#) \*const p)
- static [INLINE PURE vect\\_t loadu](#) (const [scalar\\_t](#) \*const p)
- static [INLINE void store](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)
- static [INLINE void storeu](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)
- static [INLINE void stream](#) ([scalar\\_t](#) \*p, const [vect\\_t](#) v)
- template<int s>  
static [INLINE CONST vect\\_t sll](#) (const [vect\\_t](#) a)
- template<int s>  
static [INLINE CONST vect\\_t srl](#) (const [vect\\_t](#) a)
- template<int s>  
static [INLINE CONST vect\\_t sra](#) (const [vect\\_t](#) a)
- template<uint8\_t s>  
static [INLINE CONST vect\\_t shuffle](#) (const [vect\\_t](#) a)
- static [INLINE CONST vect\\_t unpacklo\\_intrinsic](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpackhi\\_intrinsic](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpacklo](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpackhi](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE void unpacklohi](#) ([vect\\_t](#) &lo, [vect\\_t](#) &hi, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t pack\\_even](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t pack\\_odd](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)

- static `INLINE` void `pack` (`vect_t` &even, `vect_t` &odd, const `vect_t` a, const `vect_t` b)
- static `INLINE` void `transpose` (`vect_t` &r0, `vect_t` &r1, `vect_t` &r2, `vect_t` &r3)
- template<uint8\_t s>
  - static `INLINE` `CONST` `vect_t` `blend` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `add` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `addin` (`vect_t` &a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `sub` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `subin` (`vect_t` &a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `mullo` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `mul` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `mulhi` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `mulx` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `fmadd` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `fmaddin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `fmaddx` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `fmaddxin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `fnmadd` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `fnmaddin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `fnmaddx` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `fnmaddxin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `fmsub` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `fmsubin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `fmsubx` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `fmsubxin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `greater` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `lesser` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `greater_eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `lesser_eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `scalar_t` `hadd_to_scal` (const `vect_t` a)
- static `INLINE` `CONST` `vect_t` `round` (const `vect_t` a)
- static `INLINE` `vect_t` `mod` (`vect_t` &C, const `vect_t` &P, const `vect_t` &INVP, const `vect_t` &NEGP, const `vect_t` &MIN, const `vect_t` &MAX, `vect_t` &Q, `vect_t` &T)
- static `INLINE` `CONST` `vect_t` `zero` ()
- template<uint8\_t s>
  - static `INLINE` `CONST` `vect_t` `sll128` (const `vect_t` a)
- template<uint8\_t s>
  - static `INLINE` `CONST` `vect_t` `srl128` (const `vect_t` a)
- static `INLINE` `CONST` `vect_t` `vand` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `vor` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `vxor` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `vandnot` (const `vect_t` a, const `vect_t` b)

### Static Public Attributes

- static const constexpr size\_t `vect_size` = 4
- static const constexpr size\_t `alignment` = 16

## 12.281.1 Member Typedef Documentation

### 12.281.1.1 `vect_t`

```
using vect_t = __m128i
```

### 12.281.1.2 `scalar_t`

```
using scalar_t = int32_t
```

**12.281.1.3 aligned\_allocator**

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

**12.281.1.4 aligned\_vector**

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

**12.281.1.5 is\_same\_element**

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

**12.281.2 Member Function Documentation****12.281.2.1 type\_string()**

```
static const std::string type_string () [inline], [static]
```

**12.281.2.2 valid()**

```
template<class T >
static constexpr bool valid (
    T * p) [inline], [static], [constexpr]
```

**12.281.2.3 compliant()**

```
template<class T >
static constexpr bool compliant (
    T n) [inline], [static], [constexpr]
```

**12.281.2.4 set1()**

```
static INLINE CONST vect_t set1 (
    const scalar_t x) [inline], [static]
```

**12.281.2.5 set()**

```
static INLINE CONST vect_t set (
    const scalar_t x0,
    const scalar_t x1,
    const scalar_t x2,
    const scalar_t x3) [inline], [static]
```

**12.281.2.6 gather()**

```
template<class T >
static INLINE PURE vect_t gather (
    const scalar_t *const p,
    const T *const idx) [inline], [static]
```

**12.281.2.7 load()**

```
static INLINE PURE vect_t load (
    const scalar_t *const p) [inline], [static]
```

**12.281.2.8 loadu()**

```
static INLINE PURE vect_t loadu (
    const scalar_t *const p) [inline], [static]
```

**12.281.2.9 store()**

```
static INLINE void store (  
    scalar_t * p,  
    vect_t v) [inline], [static]
```

**12.281.2.10 storeu()**

```
static INLINE void storeu (  
    scalar_t * p,  
    vect_t v) [inline], [static]
```

**12.281.2.11 stream()**

```
static INLINE void stream (  
    scalar_t * p,  
    const vect_t v) [inline], [static]
```

**12.281.2.12 sll()**

```
template<int s>  
static INLINE CONST vect_t sll (  
    const vect_t a) [inline], [static]
```

**12.281.2.13 srl()**

```
template<int s>  
static INLINE CONST vect_t srl (  
    const vect_t a) [inline], [static]
```

**12.281.2.14 sra()**

```
template<int s>  
static INLINE CONST vect_t sra (  
    const vect_t a) [inline], [static]
```

**12.281.2.15 shuffle()**

```
template<uint8_t s>  
static INLINE CONST vect_t shuffle (  
    const vect_t a) [inline], [static]
```

**12.281.2.16 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.17 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.18 unpacklo()**

```
static INLINE CONST vect_t unpacklo (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.19 unpackhi()**

```
static INLINE CONST vect_t unpackhi (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.20 unpacklohi()**

```
static INLINE void unpacklohi (  
    vect_t & lo,  
    vect_t & hi,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.21 pack\_even()**

```
static INLINE CONST vect_t pack_even (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.22 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.23 pack()**

```
static INLINE void pack (  
    vect_t & even,  
    vect_t & odd,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.24 transpose()**

```
static INLINE void transpose (  
    vect_t & r0,  
    vect_t & r1,  
    vect_t & r2,  
    vect_t & r3) [inline], [static]
```

**12.281.2.25 blend()**

```
template<uint8_t s>  
static INLINE CONST vect_t blend (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.26 add()**

```
static INLINE CONST vect_t add (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.27 addin()**

```
static INLINE vect_t addin (  
    vect_t & a,  
    const vect_t b) [inline], [static]
```

**12.281.2.28 sub()**

```
static INLINE CONST vect_t sub (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.29 subin()**

```
static INLINE vect_t subin (  
    vect_t & a,  
    const vect_t b) [inline], [static]
```

**12.281.2.30 mullo()**

```
static INLINE CONST vect_t mullo (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.31 mul()**

```
static INLINE CONST vect_t mul (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.32 mulhi()**

```
static INLINE CONST vect_t mulhi (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.33 mulx()**

```
static INLINE CONST vect_t mulx (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.34 fmadd()**

```
static INLINE CONST vect_t fmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.35 fmaddin()**

```
static INLINE vect_t fmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.36 fmaddx()**

```
static INLINE CONST vect_t fmaddx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.37 fmaddxin()**

```
static INLINE vect_t fmaddxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.38 fnmadd()**

```
static INLINE CONST vect_t fnmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.39 fnmaddin()**

```
static INLINE vect_t fnmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.40 fnmaddx()**

```
static INLINE CONST vect_t fnmaddx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.41 fnmaddxin()**

```
static INLINE vect_t fnmaddxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.42 fmsub()**

```
static INLINE CONST vect_t fmsub (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.43 fmsubin()**

```
static INLINE vect_t fmsubin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.44 fmsubx()**

```
static INLINE CONST vect_t fmsubx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.281.2.45 fmsubxin()**

```
static INLINE vect_t fmsubxin (  
    vect_t & c,
```



```
const vect_t a,
const vect_t b) [inline], [static]
```

**12.281.2.46 eq()**

```
static INLINE CONST vect_t eq (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.281.2.47 greater()**

```
static INLINE CONST vect_t greater (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.281.2.48 lesser()**

```
static INLINE CONST vect_t lesser (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.281.2.49 greater\_eq()**

```
static INLINE CONST vect_t greater_eq (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.281.2.50 lesser\_eq()**

```
static INLINE CONST vect_t lesser_eq (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.281.2.51 hadd\_to\_scal()**

```
static INLINE CONST scalar_t hadd_to_scal (
    const vect_t a) [inline], [static]
```

**12.281.2.52 round()**

```
static INLINE CONST vect_t round (
    const vect_t a) [inline], [static]
```

**12.281.2.53 mod()**

```
static INLINE vect_t mod (
    vect_t & C,
    const vect_t & P,
    const vect_t & INVP,
    const vect_t & NEGP,
    const vect_t & MIN,
    const vect_t & MAX,
    vect_t & Q,
    vect_t & T) [inline], [static]
```

**12.281.2.54 zero()**

```
static INLINE CONST vect_t zero () [inline], [static], [inherited]
```

**12.281.2.55 sll128()**

```
template<uint8_t s>
static INLINE CONST vect_t sll128 (
    const vect_t a) [inline], [static], [inherited]
```

**12.281.2.56 srl128()**

```
template<uint8_t s>
static INLINE CONST vect_t srl128 (
    const vect_t a) [inline], [static], [inherited]
```

**12.281.2.57 vand()**

```
static INLINE CONST vect_t vand (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.281.2.58 vor()**

```
static INLINE CONST vect_t vor (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.281.2.59 vxor()**

```
static INLINE CONST vect_t vxor (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.281.2.60 vandnot()**

```
static INLINE CONST vect_t vandnot (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.281.3 Field Documentation****12.281.3.1 vect\_size**

```
const constexpr size_t vect_size = 4 [static], [constexpr]
```

**12.281.3.2 alignment**

```
const constexpr size_t alignment = 16 [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [simd128\\_int32.inl](#)

**12.282 Simd128\_impl< true, true, true, 8 > Struct Reference**

Inheritance diagram for Simd128\_impl< true, true, true, 8 >:



## Data Structures

- union [Converter](#)

## Public Types

- using [vect\\_t](#) = \_\_m128i
- using [scalar\\_t](#) = int64\_t
- using [aligned\\_allocator](#) = AlignedAllocator<[scalar\\_t](#), Alignment([alignment](#))>
- using [aligned\\_vector](#) = std::vector<[scalar\\_t](#), [aligned\\_allocator](#)>
- template<class [Field](#) >  
using [is\\_same\\_element](#) = std::is\_same<typename [Field::Element](#), [scalar\\_t](#)>

## Static Public Member Functions

- static const std::string [type\\_string](#) ()
- template<class T >  
static constexpr bool [valid](#) (T \*p)
- template<class T >  
static constexpr bool [compliant](#) (T n)
- static [INLINE CONST vect\\_t set1](#) (const [scalar\\_t](#) x)
- static [INLINE CONST vect\\_t set](#) (const [scalar\\_t](#) x0, const [scalar\\_t](#) x1)
- template<class T >  
static [INLINE PURE vect\\_t gather](#) (const [scalar\\_t](#) \*const p, const T \*const idx)
- template<int idx>  
static [INLINE CONST scalar\\_t get](#) ([vect\\_t](#) v)
- static [INLINE PURE vect\\_t load](#) (const [scalar\\_t](#) \*const p)
- static [INLINE PURE vect\\_t loadu](#) (const [scalar\\_t](#) \*const p)
- static [INLINE void store](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)
- static [INLINE void storeu](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)
- static [INLINE void stream](#) ([scalar\\_t](#) \*p, const [vect\\_t](#) v)
- template<int s>  
static [INLINE CONST vect\\_t sll](#) (const [vect\\_t](#) a)
- template<int s>  
static [INLINE CONST vect\\_t srl](#) (const [vect\\_t](#) a)
- template<int s>  
static [INLINE CONST vect\\_t sra](#) (const [vect\\_t](#) a)
- template<uint8\_t s>  
static [INLINE CONST vect\\_t shuffle](#) (const [vect\\_t](#) a)
- static [INLINE CONST vect\\_t unpacklo\\_intrinsic](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpackhi\\_intrinsic](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpacklo](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpackhi](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE void unpacklohi](#) ([vect\\_t](#) &lo, [vect\\_t](#) &hi, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t pack\\_even](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t pack\\_odd](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE void pack](#) ([vect\\_t](#) &even, [vect\\_t](#) &odd, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE void transpose](#) ([vect\\_t](#) &r0, [vect\\_t](#) &r1)

- `template<uint8_t s>`  
`static INLINE CONST vect_t blend (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t add (const vect_t a, const vect_t b)`
- `static INLINE vect_t addin (vect_t &a, const vect_t b)`
- `static INLINE CONST vect_t sub (const vect_t a, const vect_t b)`
- `static INLINE vect_t subin (vect_t &a, const vect_t b)`
- `static INLINE CONST vect_t mullo (const vect_t x0, const vect_t x1)`
- `static INLINE CONST vect_t mul (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mulhi (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mulx (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmaddin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmaddx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmaddxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fnmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fnmaddin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fnmaddx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fnmaddxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmsub (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmsubin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmsubx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmsubxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t greater (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t lesser (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t greater_eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t lesser_eq (const vect_t a, const vect_t b)`
- `static INLINE CONST scalar_t hadd_to_scal (const vect_t a)`
- `static INLINE CONST vect_t round (const vect_t a)`
- `static INLINE CONST vect_t mask_high ()`
- `static INLINE CONST vect_t mulhi_fast (vect_t x, vect_t y)`
- `static INLINE vect_t mod (vect_t &C, const __m128d &P, const __m128d &INVP, const __m128d &NEGP, const vect_t &POW50REM, const __m128d &MIN, const __m128d &MAX, __m128d &Q, __m128d &T)`
- `static INLINE CONST vect_t zero ()`
- `template<uint8_t s>`  
`static INLINE CONST vect_t sll128 (const vect_t a)`
- `template<uint8_t s>`  
`static INLINE CONST vect_t srl128 (const vect_t a)`
- `static INLINE CONST vect_t vand (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vor (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vxor (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vandnot (const vect_t a, const vect_t b)`

### Static Public Attributes

- `static const constexpr size_t vect_size = 2`
- `static const constexpr size_t alignment = 16`

### Static Protected Member Functions

- `static INLINE CONST vect_t signbits (const vect_t x)`

## 12.282.1 Member Typedef Documentation

### 12.282.1.1 vect\_t

using `vect_t` = `__m128i`

**12.282.1.2 scalar\_t**

```
using scalar_t = int64_t
```

**12.282.1.3 aligned\_allocator**

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

**12.282.1.4 aligned\_vector**

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

**12.282.1.5 is\_same\_element**

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

**12.282.2 Member Function Documentation****12.282.2.1 type\_string()**

```
static const std::string type_string () [inline], [static]
```

**12.282.2.2 valid()**

```
template<class T >
static constexpr bool valid (
    T * p) [inline], [static], [constexpr]
```

**12.282.2.3 compliant()**

```
template<class T >
static constexpr bool compliant (
    T n) [inline], [static], [constexpr]
```

**12.282.2.4 set1()**

```
static INLINE CONST vect_t set1 (
    const scalar_t x) [inline], [static]
```

**12.282.2.5 set()**

```
static INLINE CONST vect_t set (
    const scalar_t x0,
    const scalar_t x1) [inline], [static]
```

**12.282.2.6 gather()**

```
template<class T >
static INLINE PURE vect_t gather (
    const scalar_t *const p,
    const T *const idx) [inline], [static]
```

**12.282.2.7 get()**

```
template<int idx>
static INLINE CONST scalar_t get (
    vect_t v) [inline], [static]
```

**12.282.2.8 load()**

```
static INLINE PURE vect_t load (
    const scalar_t *const p) [inline], [static]
```

**12.282.2.9 loadu()**

```
static INLINE PURE vect_t loadu (
    const scalar_t *const p) [inline], [static]
```

**12.282.2.10 store()**

```
static INLINE void store (
    scalar_t * p,
    vect_t v) [inline], [static]
```

**12.282.2.11 storeu()**

```
static INLINE void storeu (
    scalar_t * p,
    vect_t v) [inline], [static]
```

**12.282.2.12 stream()**

```
static INLINE void stream (
    scalar_t * p,
    const vect_t v) [inline], [static]
```

**12.282.2.13 sll()**

```
template<int s>
static INLINE CONST vect_t sll (
    const vect_t a) [inline], [static]
```

**12.282.2.14 srl()**

```
template<int s>
static INLINE CONST vect_t srl (
    const vect_t a) [inline], [static]
```

**12.282.2.15 sra()**

```
template<int s>
static INLINE CONST vect_t sra (
    const vect_t a) [inline], [static]
```

**12.282.2.16 shuffle()**

```
template<uint8_t s>
static INLINE CONST vect_t shuffle (
    const vect_t a) [inline], [static]
```

**12.282.2.17 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.282.2.18 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.282.2.19 unpacklo()**

```
static INLINE CONST vect_t unpacklo (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.282.2.20 unpackhi()**

```
static INLINE CONST vect_t unpackhi (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.282.2.21 unpacklohi()**

```
static INLINE void unpacklohi (  
    vect_t & lo,  
    vect_t & hi,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.282.2.22 pack\_even()**

```
static INLINE CONST vect_t pack_even (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.282.2.23 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.282.2.24 pack()**

```
static INLINE void pack (  
    vect_t & even,  
    vect_t & odd,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.282.2.25 transpose()**

```
static INLINE void transpose (  
    vect_t & r0,  
    vect_t & r1) [inline], [static]
```

**12.282.2.26 blend()**

```
template<uint8_t s>  
static INLINE CONST vect_t blend (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.282.2.27 add()**

```
static INLINE CONST vect_t add (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.282.2.28 addin()**

```
static INLINE vect_t addin (  
    vect_t & a,  
    const vect_t b) [inline], [static]
```

**12.282.2.29 sub()**

```
static INLINE CONST vect_t sub (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.282.2.30 subin()**

```
static INLINE vect_t subin (  
    vect_t & a,  
    const vect_t b) [inline], [static]
```

**12.282.2.31 mullo()**

```
static INLINE CONST vect_t mullo (  
    const vect_t x0,  
    const vect_t x1) [inline], [static]
```

**12.282.2.32 mul()**

```
static INLINE CONST vect_t mul (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.282.2.33 mulhi()**

```
static INLINE CONST vect_t mulhi (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.282.2.34 mulx()**

```
static INLINE CONST vect_t mulx (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.282.2.35 fmadd()**

```
static INLINE CONST vect_t fmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.282.2.36 fmaddin()**

```
static INLINE vect_t fmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```



**12.282.2.37 fmaddx()**

```
static INLINE CONST vect_t fmaddx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.282.2.38 fmaddxin()**

```
static INLINE vect_t fmaddxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.282.2.39 fnmadd()**

```
static INLINE CONST vect_t fnmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.282.2.40 fnmaddin()**

```
static INLINE vect_t fnmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.282.2.41 fnmaddx()**

```
static INLINE CONST vect_t fnmaddx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.282.2.42 fnmaddxin()**

```
static INLINE vect_t fnmaddxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.282.2.43 fmsub()**

```
static INLINE CONST vect_t fmsub (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.282.2.44 fmsubin()**

```
static INLINE vect_t fmsubin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.282.2.45 fmsubx()**

```
static INLINE CONST vect_t fmsubx (  
    const vect_t c,
```

```
const vect_t a,  
const vect_t b) [inline], [static]
```

#### 12.282.2.46 fmsubxin()

```
static INLINE vect_t fmsubxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.282.2.47 eq()

```
static INLINE CONST vect_t eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.282.2.48 greater()

```
static INLINE CONST vect_t greater (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.282.2.49 lesser()

```
static INLINE CONST vect_t lesser (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.282.2.50 greater\_eq()

```
static INLINE CONST vect_t greater_eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.282.2.51 lesser\_eq()

```
static INLINE CONST vect_t lesser_eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.282.2.52 hadd\_to\_scal()

```
static INLINE CONST scalar_t hadd_to_scal (  
    const vect_t a) [inline], [static]
```

#### 12.282.2.53 round()

```
static INLINE CONST vect_t round (  
    const vect_t a) [inline], [static]
```

#### 12.282.2.54 mask\_high()

```
static INLINE CONST vect_t mask_high () [inline], [static]
```

#### 12.282.2.55 mulhi\_fast()

```
INLINE CONST vect_t mulhi_fast (  
    vect_t x,  
    vect_t y) [static]
```

**12.282.2.56 mod()**

```

INLINE vect_t mod (
    vect_t & C,
    const __m128d & P,
    const __m128d & INVP,
    const __m128d & NEGP,
    const vect_t & POW50REM,
    const __m128d & MIN,
    const __m128d & MAX,
    __m128d & Q,
    __m128d & T) [static]

```

**12.282.2.57 signbits()**

```

static INLINE CONST vect_t signbits (
    const vect_t x) [inline], [static], [protected]

```

**12.282.2.58 zero()**

```

static INLINE CONST vect_t zero () [inline], [static], [inherited]

```

**12.282.2.59 sll128()**

```

template<uint8_t s>
static INLINE CONST vect_t sll128 (
    const vect_t a) [inline], [static], [inherited]

```

**12.282.2.60 srl128()**

```

template<uint8_t s>
static INLINE CONST vect_t srl128 (
    const vect_t a) [inline], [static], [inherited]

```

**12.282.2.61 vand()**

```

static INLINE CONST vect_t vand (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]

```

**12.282.2.62 vor()**

```

static INLINE CONST vect_t vor (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]

```

**12.282.2.63 vxor()**

```

static INLINE CONST vect_t vxor (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]

```

**12.282.2.64 vandnot()**

```

static INLINE CONST vect_t vandnot (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]

```

## 12.282.3 Field Documentation

### 12.282.3.1 vect\_size

```
const constexpr size_t vect_size = 2 [static], [constexpr]
```

### 12.282.3.2 alignment

```
const constexpr size_t alignment = 16 [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [simd128\\_int64.inl](#)

## 12.283 Simd128i\_base Struct Reference

Inheritance diagram for Simd128i\_base:



### Public Types

- using [vect\\_t](#) = \_\_m128i

### Static Public Member Functions

- static [INLINE CONST vect\\_t zero](#) ()
- template<uint8\_t s>  
static [INLINE CONST vect\\_t sll128](#) (const [vect\\_t](#) a)
- template<uint8\_t s>  
static [INLINE CONST vect\\_t srl128](#) (const [vect\\_t](#) a)
- static [INLINE CONST vect\\_t vand](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t vor](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t vxor](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t vandnot](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)

## 12.283.1 Member Typedef Documentation

### 12.283.1.1 vect\_t

```
using vect\_t = __m128i
```

## 12.283.2 Member Function Documentation

### 12.283.2.1 zero()

```
static INLINE CONST vect\_t zero () [inline], [static]
```

### 12.283.2.2 sll128()

```
template<uint8_t s>
static INLINE CONST vect\_t sll128 (
    const vect\_t a) [inline], [static]
```

**12.283.2.3 srl128()**

```
template<uint8_t s>
static INLINE CONST vect_t srl128 (
    const vect_t a) [inline], [static]
```

**12.283.2.4 vand()**

```
static INLINE CONST vect_t vand (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.283.2.5 vor()**

```
static INLINE CONST vect_t vor (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.283.2.6 vxor()**

```
static INLINE CONST vect_t vxor (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.283.2.7 vandnot()**

```
static INLINE CONST vect_t vandnot (
    const vect_t a,
    const vect_t b) [inline], [static]
```

The documentation for this struct was generated from the following file:

- [simd128.inl](#)

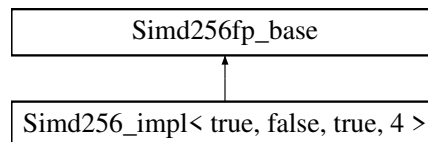
**12.284 Simd256\_impl< ArithType, Int, Signed, Size > Struct Template Reference**

The documentation for this struct was generated from the following file:

- [simd256.inl](#)

**12.285 Simd256\_impl< true, false, true, 4 > Struct Reference**

Inheritance diagram for Simd256\_impl< true, false, true, 4 >:



The documentation for this struct was generated from the following file:

- [simd256\\_float.inl](#)

## 12.286 Simd256\_impl< true, false, true, 8 > Struct Reference

Inheritance diagram for Simd256\_impl< true, false, true, 8 >:



### Data Structures

- union [Converter](#)

### Public Types

- using [vect\\_t](#) = \_\_m256d
- using [scalar\\_t](#) = double
- using [aligned\\_allocator](#) = AlignedAllocator<[scalar\\_t](#), Alignment([alignment](#))>
- using [aligned\\_vector](#) = std::vector<[scalar\\_t](#), [aligned\\_allocator](#)>
- template<class [Field](#) >  
using [is\\_same\\_element](#) = std::is\_same<typename [Field::Element](#), [scalar\\_t](#)>

### Static Public Member Functions

- static const std::string [type\\_string](#) ()
- template<class T >  
static constexpr bool [valid](#) (T \*p)
- template<class T >  
static constexpr bool [compliant](#) (T n)
- static [INLINE CONST vect\\_t zero](#) ()
- static [INLINE CONST vect\\_t set1](#) (const [scalar\\_t](#) x)
- static [INLINE CONST vect\\_t set](#) (const [scalar\\_t](#) x1, const [scalar\\_t](#) x2, const [scalar\\_t](#) x3, const [scalar\\_t](#) x4)
- template<class T >  
static [INLINE PURE vect\\_t gather](#) (const [scalar\\_t](#) \*const p, const T \*const idx)
- static [INLINE PURE vect\\_t load](#) (const [scalar\\_t](#) \*const p)
- static [INLINE PURE vect\\_t loadu](#) (const [scalar\\_t](#) \*const p)
- static [INLINE void store](#) (const [scalar\\_t](#) \*p, const [vect\\_t](#) v)
- static [INLINE void storeu](#) (const [scalar\\_t](#) \*p, const [vect\\_t](#) v)
- static [INLINE void stream](#) (const [scalar\\_t](#) \*p, const [vect\\_t](#) v)
- static [INLINE CONST vect\\_t unpacklo\\_intrinsic](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpackhi\\_intrinsic](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpacklo](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpackhi](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE void unpacklohi](#) ([vect\\_t](#) &lo, [vect\\_t](#) &hi, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t pack\\_even](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t pack\\_odd](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE void pack](#) ([vect\\_t](#) &even, [vect\\_t](#) &odd, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE void transpose](#) ([vect\\_t](#) &r0, [vect\\_t](#) &r1, [vect\\_t](#) &r2, [vect\\_t](#) &r3)
- template<uint8\_t s>  
static [INLINE CONST vect\\_t blend](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t blendv](#) (const [vect\\_t](#) a, const [vect\\_t](#) b, const [vect\\_t](#) mask)
- static [INLINE CONST vect\\_t add](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE vect\\_t addin](#) ([vect\\_t](#) &a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t sub](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t subin](#) ([vect\\_t](#) &a, const [vect\\_t](#) b)

- static `INLINE CONST vect_t mul` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t mulin` (`vect_t` &a, const `vect_t` b)
- static `INLINE CONST vect_t div` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmadd` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmaddin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fnmadd` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fnmaddin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmsub` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmsubin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t lesser` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t lesser_eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t greater` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t greater_eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t vand` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t vor` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t vxor` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t vandnot` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t floor` (const `vect_t` a)
- static `INLINE CONST vect_t ceil` (const `vect_t` a)
- static `INLINE CONST vect_t round` (const `vect_t` a)
- static `INLINE CONST vect_t hadd` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST scalar_t hadd_to_scal` (const `vect_t` a)
- static `INLINE vect_t mod` (`vect_t` &C, const `vect_t` &P, const `vect_t` &INVP, const `vect_t` &NEGP, const `vect_t` &MIN, const `vect_t` &MAX, `vect_t` &Q, `vect_t` &T)

### Static Public Attributes

- static const constexpr `size_t vect_size` = 4
- static const constexpr `size_t alignment` = 32

## 12.286.1 Member Typedef Documentation

### 12.286.1.1 vect\_t

```
using vect_t = __m256d
```

### 12.286.1.2 scalar\_t

```
using scalar_t = double
```

### 12.286.1.3 aligned\_allocator

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

### 12.286.1.4 aligned\_vector

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

### 12.286.1.5 is\_same\_element

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

## 12.286.2 Member Function Documentation

### 12.286.2.1 type\_string()

```
static const std::string type_string () [inline], [static]
```

### 12.286.2.2 valid()

```
template<class T >
static constexpr bool valid (
    T * p) [inline], [static], [constexpr]
```

### 12.286.2.3 compliant()

```
template<class T >
static constexpr bool compliant (
    T n) [inline], [static], [constexpr]
```

### 12.286.2.4 zero()

```
static INLINE CONST vect_t zero () [inline], [static]
```

### 12.286.2.5 set1()

```
static INLINE CONST vect_t set1 (
    const scalar_t x) [inline], [static]
```

### 12.286.2.6 set()

```
static INLINE CONST vect_t set (
    const scalar_t x1,
    const scalar_t x2,
    const scalar_t x3,
    const scalar_t x4) [inline], [static]
```

### 12.286.2.7 gather()

```
template<class T >
static INLINE PURE vect_t gather (
    const scalar_t *const p,
    const T *const idx) [inline], [static]
```

### 12.286.2.8 load()

```
static INLINE PURE vect_t load (
    const scalar_t *const p) [inline], [static]
```

### 12.286.2.9 loadu()

```
static INLINE PURE vect_t loadu (
    const scalar_t *const p) [inline], [static]
```

### 12.286.2.10 store()

```
static INLINE void store (
    const scalar_t * p,
    const vect_t v) [inline], [static]
```

### 12.286.2.11 storeu()

```
static INLINE void storeu (
    const scalar_t * p,
    const vect_t v) [inline], [static]
```



**12.286.2.12 stream()**

```
static INLINE void stream (  
    const scalar_t * p,  
    const vect_t v) [inline], [static]
```

**12.286.2.13 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.286.2.14 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.286.2.15 unpacklo()**

```
static INLINE CONST vect_t unpacklo (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.286.2.16 unpackhi()**

```
static INLINE CONST vect_t unpackhi (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.286.2.17 unpacklohi()**

```
static INLINE void unpacklohi (  
    vect_t & lo,  
    vect_t & hi,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.286.2.18 pack\_even()**

```
static INLINE CONST vect_t pack_even (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.286.2.19 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.286.2.20 pack()**

```
static INLINE void pack (  
    vect_t & even,  
    vect_t & odd,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.286.2.21 transpose()

```
static INLINE void transpose (  
    vect_t & r0,  
    vect_t & r1,  
    vect_t & r2,  
    vect_t & r3) [inline], [static]
```

#### 12.286.2.22 blend()

```
template<uint8_t s>  
static INLINE CONST vect_t blend (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.286.2.23 blendv()

```
static INLINE CONST vect_t blendv (  
    const vect_t a,  
    const vect_t b,  
    const vect_t mask) [inline], [static]
```

#### 12.286.2.24 add()

```
static INLINE CONST vect_t add (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.286.2.25 addin()

```
static INLINE vect_t addin (  
    vect_t & a,  
    const vect_t b) [inline], [static]
```

#### 12.286.2.26 sub()

```
static INLINE CONST vect_t sub (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.286.2.27 subin()

```
static INLINE CONST vect_t subin (  
    vect_t & a,  
    const vect_t b) [inline], [static]
```

#### 12.286.2.28 mul()

```
static INLINE CONST vect_t mul (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.286.2.29 mulin()

```
static INLINE CONST vect_t mulin (  
    vect_t & a,  
    const vect_t b) [inline], [static]
```

**12.286.2.30 div()**

```
static INLINE CONST vect_t div (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.286.2.31 fmadd()**

```
static INLINE CONST vect_t fmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.286.2.32 fmaddin()**

```
static INLINE CONST vect_t fmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.286.2.33 fnmadd()**

```
static INLINE CONST vect_t fnmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.286.2.34 fnmaddin()**

```
static INLINE CONST vect_t fnmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.286.2.35 fmsub()**

```
static INLINE CONST vect_t fmsub (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.286.2.36 fmsubin()**

```
static INLINE CONST vect_t fmsubin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.286.2.37 eq()**

```
static INLINE CONST vect_t eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.286.2.38 lesser()**

```
static INLINE CONST vect_t lesser (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.286.2.39 lesser\_eq()**

```
static INLINE CONST vect_t lesser_eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.286.2.40 greater()**

```
static INLINE CONST vect_t greater (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.286.2.41 greater\_eq()**

```
static INLINE CONST vect_t greater_eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.286.2.42 vand()**

```
static INLINE CONST vect_t vand (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.286.2.43 vor()**

```
static INLINE CONST vect_t vor (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.286.2.44 vxor()**

```
static INLINE CONST vect_t vxor (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.286.2.45 vandnot()**

```
static INLINE CONST vect_t vandnot (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.286.2.46 floor()**

```
static INLINE CONST vect_t floor (  
    const vect_t a) [inline], [static]
```

**12.286.2.47 ceil()**

```
static INLINE CONST vect_t ceil (  
    const vect_t a) [inline], [static]
```

**12.286.2.48 round()**

```
static INLINE CONST vect_t round (  
    const vect_t a) [inline], [static]
```

**12.286.2.49 hadd()**

```
static INLINE CONST vect_t hadd (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.286.2.50 hadd\_to\_scal()**

```
static INLINE CONST scalar_t hadd_to_scal (
    const vect_t a) [inline], [static]
```

**12.286.2.51 mod()**

```
static INLINE vect_t mod (
    vect_t & C,
    const vect_t & P,
    const vect_t & INV_P,
    const vect_t & NEG_P,
    const vect_t & MIN,
    const vect_t & MAX,
    vect_t & Q,
    vect_t & T) [inline], [static]
```

**12.286.3 Field Documentation****12.286.3.1 vect\_size**

```
const constexpr size_t vect_size = 4 [static], [constexpr]
```

**12.286.3.2 alignment**

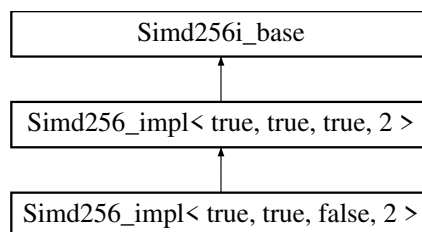
```
const constexpr size_t alignment = 32 [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [simd256\\_double.inl](#)

**12.287 Simd256\_impl< true, true, false, 2 > Struct Reference**

Inheritance diagram for Simd256\_impl< true, true, false, 2 >:

**Data Structures**

- union [Converter](#)

**Public Types**

- using [scalar\\_t](#) = uint16\_t
- using [aligned\\_allocator](#) = AlignedAllocator<[scalar\\_t](#), Alignment([alignment](#))>
- using [aligned\\_vector](#) = std::vector<[scalar\\_t](#), [aligned\\_allocator](#)>

- `template<class Field >`  
`using is_same_element = std::is_same<typename Field::Element, scalar_t>`
- `using simdHalf = Simd128<scalar_t>`
- `using vect_t = __m256i`
- `using half_t = __m128i`

### Static Public Member Functions

- `static const std::string type_string ()`
- `static INLINE CONST vect_t set1 (const scalar_t x)`
- `static INLINE CONST vect_t set (const scalar_t x0, const scalar_t x1, const scalar_t x2, const scalar_t x3, const scalar_t x4, const scalar_t x5, const scalar_t x6, const scalar_t x7, const scalar_t x8, const scalar_t x9, const scalar_t x10, const scalar_t x11, const scalar_t x12, const scalar_t x13, const scalar_t x14, const scalar_t x15)`
- `template<class T >`  
`static INLINE PURE vect_t gather (const scalar_t *const p, const T *const idx)`
- `static INLINE PURE vect_t load (const scalar_t *const p)`
- `static INLINE PURE vect_t loadu (const scalar_t *const p)`
- `static INLINE void store (scalar_t *p, vect_t v)`
- `static INLINE void storeu (scalar_t *p, vect_t v)`
- `static INLINE void stream (scalar_t *p, const vect_t v)`
- `template<int s>`  
`static INLINE CONST vect_t sra (const vect_t a)`
- `static INLINE CONST vect_t greater (vect_t a, vect_t b)`
- `static INLINE CONST vect_t lesser (vect_t a, vect_t b)`
- `static INLINE CONST vect_t greater_eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t lesser_eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mulhi (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mulx (vect_t a, vect_t b)`
- `static INLINE CONST vect_t fmaddx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmaddxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fnmaddx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fnmaddxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmsubx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmsubxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST scalar_t hadd_to_scal (const vect_t a)`
- `template<class T >`  
`static constexpr bool valid (T *p)`
- `template<class T >`  
`static constexpr bool compliant (T n)`
- `static INLINE CONST vect_t set1 (const scalar_t x)`
- `static INLINE CONST vect_t set (const scalar_t x0, const scalar_t x1, const scalar_t x2, const scalar_t x3, const scalar_t x4, const scalar_t x5, const scalar_t x6, const scalar_t x7, const scalar_t x8, const scalar_t x9, const scalar_t x10, const scalar_t x11, const scalar_t x12, const scalar_t x13, const scalar_t x14, const scalar_t x15)`
- `template<class T >`  
`static INLINE PURE vect_t gather (const scalar_t *const p, const T *const idx)`
- `static INLINE PURE vect_t load (const scalar_t *const p)`
- `static INLINE PURE vect_t loadu (const scalar_t *const p)`
- `static INLINE void store (scalar_t *p, vect_t v)`
- `static INLINE void storeu (scalar_t *p, vect_t v)`
- `static INLINE void stream (scalar_t *p, const vect_t v)`
- `template<int s>`  
`static INLINE CONST vect_t sll (const vect_t a)`
- `template<int s>`  
`static INLINE CONST vect_t srl (const vect_t a)`

- `template<uint64_t s>`  
static `INLINE CONST vect_t shuffle` (const `vect_t` a)
- static `INLINE CONST vect_t unpacklo_intrinsic` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t unpackhi_intrinsic` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t unpacklo` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t unpackhi` (const `vect_t` a, const `vect_t` b)
- static `INLINE void unpacklohi` (`vect_t` &lo, `vect_t` &hi, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t pack_even` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t pack_odd` (const `vect_t` a, const `vect_t` b)
- static `INLINE void pack` (`vect_t` &even, `vect_t` &odd, const `vect_t` a, const `vect_t` b)
- static `INLINE void transpose` (`vect_t` &r0, `vect_t` &r1, `vect_t` &r2, `vect_t` &r3, `vect_t` &r4, `vect_t` &r5, `vect_t` &r6, `vect_t` &r7, `vect_t` &r8, `vect_t` &r9, `vect_t` &r10, `vect_t` &r11, `vect_t` &r12, `vect_t` &r13, `vect_t` &r14, `vect_t` &r15)
- `template<uint16_t s, typename std::enable_if<(s & 0x00ff) == (s > > 8)>::type * = nullptr>`  
static `INLINE vect_t blend` (const `vect_t` a, const `vect_t` b)
- `template<uint16_t s, typename std::enable_if<(s & 0x00ff) != (s > > 8)>::type * = nullptr>`  
static `INLINE vect_t blend` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t add` (const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t addin` (`vect_t` &a, const `vect_t` b)
- static `INLINE CONST vect_t sub` (const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t subin` (`vect_t` &a, const `vect_t` b)
- static `INLINE CONST vect_t mullo` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t mul` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmadd` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fmadin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fnmadd` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fnmadin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmsub` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fmsubin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t round` (const `vect_t` a)
- static `INLINE vect_t mod` (`vect_t` &C, const `vect_t` &P, const `vect_t` &INVP, const `vect_t` &NEGP, const `vect_t` &MIN, const `vect_t` &MAX, `vect_t` &Q, `vect_t` &T)
- static `INLINE CONST vect_t zero` ()

### Static Public Attributes

- static const constexpr size\_t `vect_size` = 16
- static const constexpr size\_t `alignment` = 32

## 12.287.1 Member Typedef Documentation

### 12.287.1.1 scalar\_t

```
using scalar_t = uint16_t
```

### 12.287.1.2 aligned\_allocator

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

### 12.287.1.3 aligned\_vector

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

### 12.287.1.4 is\_same\_element

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

**12.287.1.5 simdHalf**

```
using simdHalf = Simd128<scalar_t>
```

**12.287.1.6 vect\_t**

```
using vect_t = __m256i [inherited]
```

**12.287.1.7 half\_t**

```
using half_t = __m128i [inherited]
```

**12.287.2 Member Function Documentation****12.287.2.1 type\_string()**

```
static const std::string type_string () [inline], [static]
```

**12.287.2.2 set1() [1/2]**

```
static INLINE CONST vect_t set1 (
    const scalar_t x) [inline], [static]
```

**12.287.2.3 set() [1/2]**

```
static INLINE CONST vect_t set (
    const scalar_t x0,
    const scalar_t x1,
    const scalar_t x2,
    const scalar_t x3,
    const scalar_t x4,
    const scalar_t x5,
    const scalar_t x6,
    const scalar_t x7,
    const scalar_t x8,
    const scalar_t x9,
    const scalar_t x10,
    const scalar_t x11,
    const scalar_t x12,
    const scalar_t x13,
    const scalar_t x14,
    const scalar_t x15) [inline], [static]
```

**12.287.2.4 gather() [1/2]**

```
template<class T >
static INLINE PURE vect_t gather (
    const scalar_t *const p,
    const T *const idx) [inline], [static]
```

**12.287.2.5 load() [1/2]**

```
static INLINE PURE vect_t load (
    const scalar_t *const p) [inline], [static]
```

**12.287.2.6 loadu() [1/2]**

```
static INLINE PURE vect_t loadu (
    const scalar_t *const p) [inline], [static]
```



**12.287.2.7 store()** [1/2]

```
static INLINE void store (  
    scalar_t * p,  
    vect_t v) [inline], [static]
```

**12.287.2.8 storeu()** [1/2]

```
static INLINE void storeu (  
    scalar_t * p,  
    vect_t v) [inline], [static]
```

**12.287.2.9 stream()** [1/2]

```
static INLINE void stream (  
    scalar_t * p,  
    const vect_t v) [inline], [static]
```

**12.287.2.10 sra()**

```
template<int s>  
static INLINE CONST vect_t sra (  
    const vect_t a) [inline], [static]
```

**12.287.2.11 greater()**

```
static INLINE CONST vect_t greater (  
    vect_t a,  
    vect_t b) [inline], [static]
```

**12.287.2.12 lesser()**

```
static INLINE CONST vect_t lesser (  
    vect_t a,  
    vect_t b) [inline], [static]
```

**12.287.2.13 greater\_eq()**

```
static INLINE CONST vect_t greater_eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.287.2.14 lesser\_eq()**

```
static INLINE CONST vect_t lesser_eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.287.2.15 mulhi()**

```
static INLINE CONST vect_t mulhi (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.287.2.16 mulx()**

```
static INLINE CONST vect_t mulx (  
    vect_t a,  
    vect_t b) [inline], [static]
```

**12.287.2.17 fmaddx()**

```
static INLINE CONST vect_t fmaddx (
    const vect_t c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.287.2.18 fmaddxin()**

```
static INLINE vect_t fmaddxin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.287.2.19 fnmaddx()**

```
static INLINE CONST vect_t fnmaddx (
    const vect_t c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.287.2.20 fnmaddxin()**

```
static INLINE vect_t fnmaddxin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.287.2.21 fmsubx()**

```
static INLINE CONST vect_t fmsubx (
    const vect_t c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.287.2.22 fmsubxin()**

```
static INLINE vect_t fmsubxin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.287.2.23 hadd\_to\_scal()**

```
static INLINE CONST scalar_t hadd_to_scal (
    const vect_t a) [inline], [static]
```

**12.287.2.24 valid()**

```
template<class T >
static constexpr bool valid (
    T * p) [inline], [static], [constexpr], [inherited]
```

**12.287.2.25 compliant()**

```
template<class T >
static constexpr bool compliant (
    T n) [inline], [static], [constexpr], [inherited]
```

**12.287.2.26 set1() [2/2]**

```
static INLINE CONST vect_t set1 (  
    const scalar_t x) [inline], [static], [inherited]
```

**12.287.2.27 set() [2/2]**

```
static INLINE CONST vect_t set (  
    const scalar_t x0,  
    const scalar_t x1,  
    const scalar_t x2,  
    const scalar_t x3,  
    const scalar_t x4,  
    const scalar_t x5,  
    const scalar_t x6,  
    const scalar_t x7,  
    const scalar_t x8,  
    const scalar_t x9,  
    const scalar_t x10,  
    const scalar_t x11,  
    const scalar_t x12,  
    const scalar_t x13,  
    const scalar_t x14,  
    const scalar_t x15) [inline], [static], [inherited]
```

**12.287.2.28 gather() [2/2]**

```
template<class T >  
static INLINE PURE vect_t gather (  
    const scalar_t *const p,  
    const T *const idx) [inline], [static], [inherited]
```

**12.287.2.29 load() [2/2]**

```
static INLINE PURE vect_t load (  
    const scalar_t *const p) [inline], [static], [inherited]
```

**12.287.2.30 loadu() [2/2]**

```
static INLINE PURE vect_t loadu (  
    const scalar_t *const p) [inline], [static], [inherited]
```

**12.287.2.31 store() [2/2]**

```
static INLINE void store (  
    scalar_t * p,  
    vect_t v) [inline], [static], [inherited]
```

**12.287.2.32 storeu() [2/2]**

```
static INLINE void storeu (  
    scalar_t * p,  
    vect_t v) [inline], [static], [inherited]
```

**12.287.2.33 stream() [2/2]**

```
static INLINE void stream (  
    scalar_t * p,  
    const vect_t v) [inline], [static], [inherited]
```

**12.287.2.34 sll()**

```
template<int s>
static INLINE CONST vect_t sll (
    const vect_t a) [inline], [static], [inherited]
```

**12.287.2.35 srl()**

```
template<int s>
static INLINE CONST vect_t srl (
    const vect_t a) [inline], [static], [inherited]
```

**12.287.2.36 shuffle()**

```
template<uint64_t s>
static INLINE CONST vect_t shuffle (
    const vect_t a) [inline], [static], [inherited]
```

**12.287.2.37 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.287.2.38 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.287.2.39 unpacklo()**

```
static INLINE CONST vect_t unpacklo (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.287.2.40 unpackhi()**

```
static INLINE CONST vect_t unpackhi (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.287.2.41 unpacklohi()**

```
static INLINE void unpacklohi (
    vect_t & lo,
    vect_t & hi,
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.287.2.42 pack\_even()**

```
static INLINE CONST vect_t pack_even (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.287.2.43 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.287.2.44 pack()**

```
static INLINE void pack (
    vect_t & even,
    vect_t & odd,
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.287.2.45 transpose()**

```
static INLINE void transpose (
    vect_t & r0,
    vect_t & r1,
    vect_t & r2,
    vect_t & r3,
    vect_t & r4,
    vect_t & r5,
    vect_t & r6,
    vect_t & r7,
    vect_t & r8,
    vect_t & r9,
    vect_t & r10,
    vect_t & r11,
    vect_t & r12,
    vect_t & r13,
    vect_t & r14,
    vect_t & r15) [inline], [static], [inherited]
```

**12.287.2.46 blend() [1/2]**

```
template<uint16_t s, typename std::enable_if<(s &0x00ff)==(s > > 8)>::type * = nullptr>
static INLINE vect_t blend (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.287.2.47 blend() [2/2]**

```
template<uint16_t s, typename std::enable_if<(s &0x00ff) !=(s > > 8)>::type * = nullptr>
static INLINE vect_t blend (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.287.2.48 add()**

```
static INLINE CONST vect_t add (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.287.2.49 addin()**

```
static INLINE vect_t addin (
    vect_t & a,
    const vect_t b) [inline], [static], [inherited]
```

**12.287.2.50 sub()**

```
static INLINE CONST vect_t sub (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.287.2.51 subin()**

```
static INLINE vect_t subin (  
    vect_t & a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.287.2.52 mullo()**

```
static INLINE CONST vect_t mullo (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.287.2.53 mul()**

```
static INLINE CONST vect_t mul (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.287.2.54 fmadd()**

```
static INLINE CONST vect_t fmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.287.2.55 fmaddin()**

```
static INLINE vect_t fmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.287.2.56 fnmadd()**

```
static INLINE CONST vect_t fnmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.287.2.57 fnmaddin()**

```
static INLINE vect_t fnmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.287.2.58 fmsub()**

```
static INLINE CONST vect_t fmsub (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.287.2.59 fmsubin()**

```
static INLINE vect_t fmsubin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.287.2.60 eq()**

```
static INLINE CONST vect_t eq (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.287.2.61 round()**

```
static INLINE CONST vect_t round (
    const vect_t a) [inline], [static], [inherited]
```

**12.287.2.62 mod()**

```
static INLINE vect_t mod (
    vect_t & C,
    const vect_t & P,
    const vect_t & INVP,
    const vect_t & NEGP,
    const vect_t & MIN,
    const vect_t & MAX,
    vect_t & Q,
    vect_t & T) [inline], [static], [inherited]
```

**12.287.2.63 zero()**

```
static INLINE CONST vect_t zero () [inline], [static], [inherited]
```

**12.287.3 Field Documentation****12.287.3.1 vect\_size**

```
const constexpr size_t vect_size = 16 [static], [constexpr], [inherited]
```

**12.287.3.2 alignment**

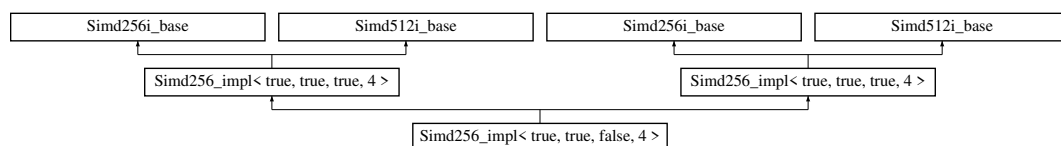
```
const constexpr size_t alignment = 32 [static], [constexpr], [inherited]
```

The documentation for this struct was generated from the following file:

- [simd256\\_int16.inl](#)

**12.288 Simd256\_impl< true, true, false, 4 > Struct Reference**

Inheritance diagram for Simd256\_impl< true, true, false, 4 >:

**Data Structures**

- union [Converter](#)

**Public Types**

- using `scalar_t` = `uint32_t`
- using `aligned_allocator` = `AlignedAllocator<scalar_t, Alignment(alignment)>`
- using `aligned_vector` = `std::vector<scalar_t, aligned_allocator>`

- `template<class Field >`  
  using `is_same_element` = `std::is_same<typename Field::Element, scalar_t>`
- using `simdHalf` = `Simd128<scalar_t>`
- using `vect_t` = `__m256i`
- using `vect_t` = `__m512i`
- using `half_t` = `__m128i`
- using `half_t` = `__m256i`

## Public Member Functions

- `template<class Field >`  
  using `is_same_element`

## Static Public Member Functions

- static const `std::string` `type_string` ()
- static `INLINE CONST vect_t` `set1` (const `scalar_t` x)
- static `INLINE CONST vect_t` `set` (const `scalar_t` x0, const `scalar_t` x1, const `scalar_t` x2, const `scalar_t` x3, const `scalar_t` x4, const `scalar_t` x5, const `scalar_t` x6, const `scalar_t` x7)
- `template<class T >`  
  static `INLINE PURE vect_t` `gather` (const `scalar_t` \*const p, const T \*const idx)
- static `INLINE PURE vect_t` `load` (const `scalar_t` \*const p)
- static `INLINE PURE vect_t` `loadu` (const `scalar_t` \*const p)
- static `INLINE` void `store` (`scalar_t` \*p, `vect_t` v)
- static `INLINE` void `storeu` (`scalar_t` \*p, `vect_t` v)
- static `INLINE` void `stream` (`scalar_t` \*p, const `vect_t` v)
- `template<int s>`  
  static `INLINE CONST vect_t` `sra` (const `vect_t` a)
- static `INLINE CONST vect_t` `greater` (`vect_t` a, `vect_t` b)
- static `INLINE CONST vect_t` `lesser` (`vect_t` a, `vect_t` b)
- static `INLINE CONST vect_t` `greater_eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t` `lesser_eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t` `mulhi` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t` `mulx` (`vect_t` a, `vect_t` b)
- static `INLINE CONST vect_t` `fmaddx` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t` `fmaddxin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t` `fnmaddx` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t` `fnmaddxin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t` `fmsubx` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t` `fmsubxin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST scalar_t` `hadd_to_scal` (const `vect_t` a)
- static const `std::string` `type_string` ()
- static `INLINE CONST vect_t` `set1` (const `scalar_t` x)
- static `INLINE CONST vect_t` `set` (const `scalar_t` x0, const `scalar_t` x1, const `scalar_t` x2, const `scalar_t` x3, const `scalar_t` x4, const `scalar_t` x5, const `scalar_t` x6, const `scalar_t` x7)
- `template<class T >`  
  static `INLINE PURE vect_t` `gather` (const `scalar_t` \*const p, const T \*const idx)
- static `INLINE PURE vect_t` `load` (const `scalar_t` \*const p)
- static `INLINE PURE vect_t` `loadu` (const `scalar_t` \*const p)
- static `INLINE` void `store` (`scalar_t` \*p, `vect_t` v)
- static `INLINE` void `storeu` (`scalar_t` \*p, `vect_t` v)
- static `INLINE` void `stream` (`scalar_t` \*p, const `vect_t` v)
- `template<int s>`  
  static `INLINE CONST vect_t` `sra` (const `vect_t` a)
- static `INLINE CONST vect_t` `greater` (`vect_t` a, `vect_t` b)
- static `INLINE CONST vect_t` `lesser` (`vect_t` a, `vect_t` b)



- static `INLINE CONST vect_t greater_eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t lesser_eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t mulhi` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t mulx` (`vect_t` a, `vect_t` b)
- static `INLINE CONST vect_t fmaddx` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fmaddxin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fnmaddx` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fnmaddxin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmsubx` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fmsubxin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST scalar_t hadd_to_scal` (const `vect_t` a)
- template<class T >  
static constexpr bool `valid` (T \*p)
- template<class T >  
static constexpr bool `valid` (T \*p)
- template<class T >  
static constexpr bool `compliant` (T n)
- template<class T >  
static constexpr bool `compliant` (T n)
- static `INLINE CONST vect_t set1` (const `scalar_t` x)
- static `INLINE CONST vect_t set` (const `scalar_t` x0, const `scalar_t` x1, const `scalar_t` x2, const `scalar_t` x3, const `scalar_t` x4, const `scalar_t` x5, const `scalar_t` x6, const `scalar_t` x7)
- static `INLINE CONST vect_t set` (const `scalar_t` x0, const `scalar_t` x1, const `scalar_t` x2, const `scalar_t` x3, const `scalar_t` x4, const `scalar_t` x5, const `scalar_t` x6, const `scalar_t` x7, const `scalar_t` x8, const `scalar_t` x9, const `scalar_t` x10, const `scalar_t` x11, const `scalar_t` x12, const `scalar_t` x13, const `scalar_t` x14, const `scalar_t` x15)
- template<class T >  
static `INLINE PURE vect_t gather` (const `scalar_t` \*const p, const T \*const idx)
- static `INLINE PURE vect_t load` (const `scalar_t` \*const p)
- static `INLINE PURE vect_t loadu` (const `scalar_t` \*const p)
- static `INLINE void store` (`scalar_t` \*p, `vect_t` v)
- static `INLINE void storeu` (`scalar_t` \*p, `vect_t` v)
- static `INLINE void stream` (`scalar_t` \*p, const `vect_t` v)
- template<int s>  
static `INLINE CONST vect_t sll` (const `vect_t` a)
- template<int s>  
static `INLINE CONST vect_t sll` (const `vect_t` a)
- template<int s>  
static `INLINE CONST vect_t srl` (const `vect_t` a)
- template<int s>  
static `INLINE CONST vect_t srl` (const `vect_t` a)
- template<uint8\_t s>  
static `INLINE CONST vect_t shuffle_twice` (const `vect_t` a)
- template<uint8\_t s>  
static `INLINE CONST vect_t shuffle_twice` (const `vect_t` a)
- template<uint32\_t s>  
static `INLINE CONST vect_t shuffle` (const `vect_t` a)
- template<uint64\_t s>  
static `INLINE CONST vect_t shuffle` (const `vect_t` a)
- static `INLINE CONST vect_t unpacklo_intrinsic` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t unpacklo_intrinsic` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t unpackhi_intrinsic` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t unpackhi_intrinsic` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t unpacklo` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t unpacklo` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t unpackhi` (const `vect_t` a, const `vect_t` b)

- static `INLINE CONST vect_t unpackhi` (const `vect_t` a, const `vect_t` b)
- static `INLINE void unpacklohi` (`vect_t` &lo, `vect_t` &hi, const `vect_t` a, const `vect_t` b)
- static `INLINE void unpacklohi` (`vect_t` &lo, `vect_t` &hi, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t pack_even` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t pack_even` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t pack_odd` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t pack_odd` (const `vect_t` a, const `vect_t` b)
- static `INLINE void pack` (`vect_t` &even, `vect_t` &odd, const `vect_t` a, const `vect_t` b)
- static `INLINE void pack` (`vect_t` &even, `vect_t` &odd, const `vect_t` a, const `vect_t` b)
- static `INLINE void transpose` (`vect_t` &r0, `vect_t` &r1, `vect_t` &r2, `vect_t` &r3, `vect_t` &r4, `vect_t` &r5, `vect_t` &r6, `vect_t` &r7)
- static `INLINE void transpose` (`vect_t` &r0, `vect_t` &r1, `vect_t` &r2, `vect_t` &r3, `vect_t` &r4, `vect_t` &r5, `vect_t` &r6, `vect_t` &r7, `vect_t` &r8, `vect_t` &r9, `vect_t` &r10, `vect_t` &r11, `vect_t` &r12, `vect_t` &r13, `vect_t` &r14, `vect_t` &r15)
- `template<uint8_t s>`  
static `INLINE CONST vect_t blend` (const `vect_t` a, const `vect_t` b)
- `template<uint16_t s>`  
static `INLINE CONST vect_t blend` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t add` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t add` (const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t addin` (`vect_t` &a, const `vect_t` b)
- static `INLINE vect_t addin` (`vect_t` &a, const `vect_t` b)
- static `INLINE CONST vect_t sub` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t sub` (const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t subin` (`vect_t` &a, const `vect_t` b)
- static `INLINE vect_t subin` (`vect_t` &a, const `vect_t` b)
- static `INLINE CONST vect_t mullo` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t mullo` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t mul` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t mul` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmaddd` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmaddd` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fmadddin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fmadddin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fnmadd` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fnmadd` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fnmaddin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fnmaddin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmsub` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmsub` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fmsubin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fmsubin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t round` (const `vect_t` a)
- static `INLINE CONST vect_t round` (const `vect_t` a)
- static `INLINE vect_t mod` (`vect_t` &C, const `vect_t` &P, const `vect_t` &INVP, const `vect_t` &NEGP, const `vect_t` &MIN, const `vect_t` &MAX, `vect_t` &Q, `vect_t` &T)
- static `INLINE vect_t mod` (`vect_t` &C, const `vect_t` &P, const `vect_t` &INVP, const `vect_t` &NEGP, const `vect_t` &MIN, const `vect_t` &MAX, `vect_t` &Q, `vect_t` &T)
- static `INLINE CONST vect_t zero` ()
- static `INLINE CONST vect_t zero` ()
- static `INLINE CONST vect_t vor` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t vxor` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t vand` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t vandnot` (const `vect_t` a, const `vect_t` b)

**Static Public Attributes**

- static const constexpr size\_t `vect_size` = 8
- static const constexpr size\_t `alignment` = 32

**12.288.1 Member Typedef Documentation****12.288.1.1 scalar\_t**

```
typedef uint32_t scalar_t = uint32_t
```

**12.288.1.2 aligned\_allocator**

```
typedef AlignedAllocator< scalar_t, Alignment(alignment)> aligned_allocator = AlignedAllocator<scalar_t,
Alignment(alignment)>
```

**12.288.1.3 aligned\_vector**

```
typedef std::vector< scalar_t, aligned_allocator > aligned_vector = std::vector<scalar_t,
aligned_allocator>
```

**12.288.1.4 is\_same\_element**

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

**12.288.1.5 simdHalf**

```
typedef Simd128< scalar_t > simdHalf = Simd128<scalar_t>
```

**12.288.1.6 vect\_t [1/2]**

```
using vect_t = __m256i [inherited]
```

**12.288.1.7 vect\_t [2/2]**

```
using vect_t = __m512i [inherited]
```

**12.288.1.8 half\_t [1/2]**

```
using half_t = __m128i [inherited]
```

**12.288.1.9 half\_t [2/2]**

```
using half_t = __m256i [inherited]
```

**12.288.2 Member Function Documentation****12.288.2.1 type\_string() [1/2]**

```
static const std::string type_string () [inline], [static]
```

**12.288.2.2 set1() [1/3]**

```
static INLINE CONST vect_t set1 (
    const scalar_t x) [inline], [static]
```

**12.288.2.3 set()** [1/4]

```
static INLINE CONST vect_t set (
    const scalar_t x0,
    const scalar_t x1,
    const scalar_t x2,
    const scalar_t x3,
    const scalar_t x4,
    const scalar_t x5,
    const scalar_t x6,
    const scalar_t x7) [inline], [static]
```

**12.288.2.4 gather()** [1/3]

```
template<class T >
static INLINE PURE vect_t gather (
    const scalar_t *const p,
    const T *const idx) [inline], [static]
```

**12.288.2.5 load()** [1/3]

```
static INLINE PURE vect_t load (
    const scalar_t *const p) [inline], [static]
```

**12.288.2.6 loadu()** [1/3]

```
static INLINE PURE vect_t loadu (
    const scalar_t *const p) [inline], [static]
```

**12.288.2.7 store()** [1/3]

```
static INLINE void store (
    scalar_t * p,
    vect_t v) [inline], [static]
```

**12.288.2.8 storeu()** [1/3]

```
static INLINE void storeu (
    scalar_t * p,
    vect_t v) [inline], [static]
```

**12.288.2.9 stream()** [1/3]

```
static INLINE void stream (
    scalar_t * p,
    const vect_t v) [inline], [static]
```

**12.288.2.10 sra()** [1/2]

```
template<int s>
static INLINE CONST vect_t sra (
    const vect_t a) [inline], [static]
```

**12.288.2.11 greater()** [1/2]

```
static INLINE CONST vect_t greater (
    vect_t a,
    vect_t b) [inline], [static]
```

**12.288.2.12 lesser()** [1/2]

```
static INLINE CONST vect_t lesser (  
    vect_t a,  
    vect_t b) [inline], [static]
```

**12.288.2.13 greater\_eq()** [1/2]

```
static INLINE CONST vect_t greater_eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.288.2.14 lesser\_eq()** [1/2]

```
static INLINE CONST vect_t lesser_eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.288.2.15 mulhi()** [1/2]

```
static INLINE CONST vect_t mulhi (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.288.2.16 mulx()** [1/2]

```
static INLINE CONST vect_t mulx (  
    vect_t a,  
    vect_t b) [inline], [static]
```

**12.288.2.17 fmaddx()** [1/2]

```
static INLINE CONST vect_t fmaddx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.288.2.18 fmaddxin()** [1/2]

```
static INLINE vect_t fmaddxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.288.2.19 fnmaddx()** [1/2]

```
static INLINE CONST vect_t fnmaddx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.288.2.20 fnmaddxin()** [1/2]

```
static INLINE vect_t fnmaddxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.288.2.21 fmsubx()** [1/2]

```
static INLINE CONST vect_t fmsubx (
    const vect_t c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.288.2.22 fmsubxin()** [1/2]

```
static INLINE vect_t fmsubxin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.288.2.23 hadd\_to\_scal()** [1/2]

```
static INLINE CONST scalar_t hadd_to_scal (
    const vect_t a) [inline], [static]
```

**12.288.2.24 type\_string()** [2/2]

```
static const std::string type_string () [inline], [static]
```

**12.288.2.25 set1()** [2/3]

```
static INLINE CONST vect_t set1 (
    const scalar_t x) [inline], [static]
```

**12.288.2.26 set()** [2/4]

```
static INLINE CONST vect_t set (
    const scalar_t x0,
    const scalar_t x1,
    const scalar_t x2,
    const scalar_t x3,
    const scalar_t x4,
    const scalar_t x5,
    const scalar_t x6,
    const scalar_t x7) [inline], [static]
```

**12.288.2.27 gather()** [2/3]

```
template<class T >
static INLINE PURE vect_t gather (
    const scalar_t *const p,
    const T *const idx) [inline], [static]
```

**12.288.2.28 load()** [2/3]

```
static INLINE PURE vect_t load (
    const scalar_t *const p) [inline], [static]
```

**12.288.2.29 loadu()** [2/3]

```
static INLINE PURE vect_t loadu (
    const scalar_t *const p) [inline], [static]
```

**12.288.2.30 store()** [2/3]

```
static INLINE void store (  
    scalar_t * p,  
    vect_t v) [inline], [static]
```

**12.288.2.31 storeu()** [2/3]

```
static INLINE void storeu (  
    scalar_t * p,  
    vect_t v) [inline], [static]
```

**12.288.2.32 stream()** [2/3]

```
static INLINE void stream (  
    scalar_t * p,  
    const vect_t v) [inline], [static]
```

**12.288.2.33 sra()** [2/2]

```
template<int s>  
static INLINE CONST vect_t sra (  
    const vect_t a) [inline], [static]
```

**12.288.2.34 greater()** [2/2]

```
static INLINE CONST vect_t greater (  
    vect_t a,  
    vect_t b) [inline], [static]
```

**12.288.2.35 lesser()** [2/2]

```
static INLINE CONST vect_t lesser (  
    vect_t a,  
    vect_t b) [inline], [static]
```

**12.288.2.36 greater\_eq()** [2/2]

```
static INLINE CONST vect_t greater_eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.288.2.37 lesser\_eq()** [2/2]

```
static INLINE CONST vect_t lesser_eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.288.2.38 mulhi()** [2/2]

```
static INLINE CONST vect_t mulhi (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.288.2.39 mulx()** [2/2]

```
static INLINE CONST vect_t mulx (  
    vect_t a,  
    vect_t b) [inline], [static]
```

**12.288.2.40 fmaddx() [2/2]**

```
static INLINE CONST vect_t fmaddx (
    const vect_t c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.288.2.41 fmaddxin() [2/2]**

```
static INLINE vect_t fmaddxin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.288.2.42 fnmaddx() [2/2]**

```
static INLINE CONST vect_t fnmaddx (
    const vect_t c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.288.2.43 fnmaddxin() [2/2]**

```
static INLINE vect_t fnmaddxin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.288.2.44 fmsubx() [2/2]**

```
static INLINE CONST vect_t fmsubx (
    const vect_t c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.288.2.45 fmsubxin() [2/2]**

```
static INLINE vect_t fmsubxin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.288.2.46 hadd\_to\_scal() [2/2]**

```
static INLINE CONST scalar_t hadd_to_scal (
    const vect_t a) [inline], [static]
```

**12.288.2.47 is\_same\_element()**

```
template<class Field >
typedef std::is_same<typename Field::Element, scalar_t> is_same_element
```

**12.288.2.48 valid() [1/2]**

```
template<class T >
static constexpr bool valid (
    T * p) [inline], [static], [constexpr], [inherited]
```



**12.288.2.49 valid()** [2/2]

```
template<class T >
static constexpr bool valid (
    T * p) [inline], [static], [constexpr], [inherited]
```

**12.288.2.50 compliant()** [1/2]

```
template<class T >
static constexpr bool compliant (
    T n) [inline], [static], [constexpr], [inherited]
```

**12.288.2.51 compliant()** [2/2]

```
template<class T >
static constexpr bool compliant (
    T n) [inline], [static], [constexpr], [inherited]
```

**12.288.2.52 set1()** [3/3]

```
static INLINE CONST vect_t set1 (
    const scalar_t x) [inline], [static], [inherited]
```

**12.288.2.53 set()** [3/4]

```
static INLINE CONST vect_t set (
    const scalar_t x0,
    const scalar_t x1,
    const scalar_t x2,
    const scalar_t x3,
    const scalar_t x4,
    const scalar_t x5,
    const scalar_t x6,
    const scalar_t x7) [inline], [static], [inherited]
```

**12.288.2.54 set()** [4/4]

```
static INLINE CONST vect_t set (
    const scalar_t x0,
    const scalar_t x1,
    const scalar_t x2,
    const scalar_t x3,
    const scalar_t x4,
    const scalar_t x5,
    const scalar_t x6,
    const scalar_t x7,
    const scalar_t x8,
    const scalar_t x9,
    const scalar_t x10,
    const scalar_t x11,
    const scalar_t x12,
    const scalar_t x13,
    const scalar_t x14,
    const scalar_t x15) [inline], [static], [inherited]
```

**12.288.2.55 gather()** [3/3]

```
template<class T >
static INLINE PURE vect_t gather (
```

```
const scalar_t *const p,
const T *const idx) [inline], [static], [inherited]
```

#### 12.288.2.56 load() [3/3]

```
static INLINE PURE vect_t load (
    const scalar_t *const p) [inline], [static], [inherited]
```

#### 12.288.2.57 loadu() [3/3]

```
static INLINE PURE vect_t loadu (
    const scalar_t *const p) [inline], [static], [inherited]
```

#### 12.288.2.58 store() [3/3]

```
static INLINE void store (
    scalar_t * p,
    vect_t v) [inline], [static], [inherited]
```

#### 12.288.2.59 storeu() [3/3]

```
static INLINE void storeu (
    scalar_t * p,
    vect_t v) [inline], [static], [inherited]
```

#### 12.288.2.60 stream() [3/3]

```
static INLINE void stream (
    scalar_t * p,
    const vect_t v) [inline], [static], [inherited]
```

#### 12.288.2.61 sll() [1/2]

```
template<int s>
static INLINE CONST vect_t sll (
    const vect_t a) [inline], [static], [inherited]
```

#### 12.288.2.62 sll() [2/2]

```
template<int s>
static INLINE CONST vect_t sll (
    const vect_t a) [inline], [static], [inherited]
```

#### 12.288.2.63 srl() [1/2]

```
template<int s>
static INLINE CONST vect_t srl (
    const vect_t a) [inline], [static], [inherited]
```

#### 12.288.2.64 srl() [2/2]

```
template<int s>
static INLINE CONST vect_t srl (
    const vect_t a) [inline], [static], [inherited]
```

#### 12.288.2.65 shuffle\_twice() [1/2]

```
template<uint8_t s>
static INLINE CONST vect_t shuffle_twice (
    const vect_t a) [inline], [static], [inherited]
```

**12.288.2.66 shuffle\_twice()** [2/2]

```
template<uint8_t s>
static INLINE CONST vect_t shuffle_twice (
    const vect_t a) [inline], [static], [inherited]
```

**12.288.2.67 shuffle()** [1/2]

```
template<uint32_t s>
static INLINE CONST vect_t shuffle (
    const vect_t a) [inline], [static], [inherited]
```

**12.288.2.68 shuffle()** [2/2]

```
template<uint64_t s>
static INLINE CONST vect_t shuffle (
    const vect_t a) [inline], [static], [inherited]
```

**12.288.2.69 unpacklo\_intrinsic()** [1/2]

```
static INLINE CONST vect_t unpacklo_intrinsic (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.70 unpacklo\_intrinsic()** [2/2]

```
static INLINE CONST vect_t unpacklo_intrinsic (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.71 unpackhi\_intrinsic()** [1/2]

```
static INLINE CONST vect_t unpackhi_intrinsic (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.72 unpackhi\_intrinsic()** [2/2]

```
static INLINE CONST vect_t unpackhi_intrinsic (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.73 unpacklo()** [1/2]

```
static INLINE CONST vect_t unpacklo (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.74 unpacklo()** [2/2]

```
static INLINE CONST vect_t unpacklo (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.75 unpackhi()** [1/2]

```
static INLINE CONST vect_t unpackhi (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.76 unpackhi() [2/2]**

```
static INLINE CONST vect_t unpackhi (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.77 unpacklohi() [1/2]**

```
static INLINE void unpacklohi (  
    vect_t & lo,  
    vect_t & hi,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.78 unpacklohi() [2/2]**

```
static INLINE void unpacklohi (  
    vect_t & lo,  
    vect_t & hi,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.79 pack\_even() [1/2]**

```
static INLINE CONST vect_t pack_even (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.80 pack\_even() [2/2]**

```
static INLINE CONST vect_t pack_even (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.81 pack\_odd() [1/2]**

```
static INLINE CONST vect_t pack_odd (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.82 pack\_odd() [2/2]**

```
static INLINE CONST vect_t pack_odd (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.83 pack() [1/2]**

```
static INLINE void pack (  
    vect_t & even,  
    vect_t & odd,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.84 pack() [2/2]**

```
static INLINE void pack (  
    vect_t & even,  
    vect_t & odd,
```

```
const vect_t a,  
const vect_t b) [inline], [static], [inherited]
```

#### 12.288.2.85 transpose() [1/2]

```
static INLINE void transpose (  
    vect_t & r0,  
    vect_t & r1,  
    vect_t & r2,  
    vect_t & r3,  
    vect_t & r4,  
    vect_t & r5,  
    vect_t & r6,  
    vect_t & r7) [inline], [static], [inherited]
```

#### 12.288.2.86 transpose() [2/2]

```
static INLINE void transpose (  
    vect_t & r0,  
    vect_t & r1,  
    vect_t & r2,  
    vect_t & r3,  
    vect_t & r4,  
    vect_t & r5,  
    vect_t & r6,  
    vect_t & r7,  
    vect_t & r8,  
    vect_t & r9,  
    vect_t & r10,  
    vect_t & r11,  
    vect_t & r12,  
    vect_t & r13,  
    vect_t & r14,  
    vect_t & r15) [inline], [static], [inherited]
```

#### 12.288.2.87 blend() [1/2]

```
template<uint8_t s>  
static INLINE CONST vect_t blend (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

#### 12.288.2.88 blend() [2/2]

```
template<uint16_t s>  
static INLINE CONST vect_t blend (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

#### 12.288.2.89 add() [1/2]

```
static INLINE CONST vect_t add (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

#### 12.288.2.90 add() [2/2]

```
static INLINE CONST vect_t add (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.91 addin() [1/2]**

```
static INLINE vect_t addin (  
    vect_t & a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.92 addin() [2/2]**

```
static INLINE vect_t addin (  
    vect_t & a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.93 sub() [1/2]**

```
static INLINE CONST vect_t sub (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.94 sub() [2/2]**

```
static INLINE CONST vect_t sub (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.95 subin() [1/2]**

```
static INLINE vect_t subin (  
    vect_t & a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.96 subin() [2/2]**

```
static INLINE vect_t subin (  
    vect_t & a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.97 mullo() [1/2]**

```
static INLINE CONST vect_t mullo (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.98 mullo() [2/2]**

```
static INLINE CONST vect_t mullo (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.99 mul() [1/2]**

```
static INLINE CONST vect_t mul (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.100 mul() [2/2]**

```
static INLINE CONST vect_t mul (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.101 fmadd()** [1/2]

```
static INLINE CONST vect_t fmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.102 fmadd()** [2/2]

```
static INLINE CONST vect_t fmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.103 fmaddin()** [1/2]

```
static INLINE vect_t fmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.104 fmaddin()** [2/2]

```
static INLINE vect_t fmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.105 fnmadd()** [1/2]

```
static INLINE CONST vect_t fnmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.106 fnmadd()** [2/2]

```
static INLINE CONST vect_t fnmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.107 fnmaddin()** [1/2]

```
static INLINE vect_t fnmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.108 fnmaddin()** [2/2]

```
static INLINE vect_t fnmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.109 fmsub()** [1/2]

```
static INLINE CONST vect_t fmsub (  
    const vect_t c,
```

```

const vect_t a,
const vect_t b) [inline], [static], [inherited]

```

#### 12.288.2.110 fmsub() [2/2]

```

static INLINE CONST vect_t fmsub (
    const vect_t c,
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]

```

#### 12.288.2.111 fmsubin() [1/2]

```

static INLINE vect_t fmsubin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]

```

#### 12.288.2.112 fmsubin() [2/2]

```

static INLINE vect_t fmsubin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]

```

#### 12.288.2.113 eq() [1/2]

```

static INLINE CONST vect_t eq (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]

```

#### 12.288.2.114 eq() [2/2]

```

static INLINE CONST vect_t eq (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]

```

#### 12.288.2.115 round() [1/2]

```

static INLINE CONST vect_t round (
    const vect_t a) [inline], [static], [inherited]

```

#### 12.288.2.116 round() [2/2]

```

static INLINE CONST vect_t round (
    const vect_t a) [inline], [static], [inherited]

```

#### 12.288.2.117 mod() [1/2]

```

static INLINE vect_t mod (
    vect_t & C,
    const vect_t & P,
    const vect_t & INVP,
    const vect_t & NEGP,
    const vect_t & MIN,
    const vect_t & MAX,
    vect_t & Q,
    vect_t & T) [inline], [static], [inherited]

```



**12.288.2.118 mod()** [2/2]

```
static INLINE vect_t mod (
    vect_t & C,
    const vect_t & P,
    const vect_t & INVP,
    const vect_t & NEGP,
    const vect_t & MIN,
    const vect_t & MAX,
    vect_t & Q,
    vect_t & T) [inline], [static], [inherited]
```

**12.288.2.119 zero()** [1/2]

```
static INLINE CONST vect_t zero () [inline], [static], [inherited]
```

**12.288.2.120 zero()** [2/2]

```
static INLINE CONST vect_t zero () [inline], [static], [inherited]
```

**12.288.2.121 vor()**

```
static INLINE CONST vect_t vor (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.122 vxor()**

```
static INLINE CONST vect_t vxor (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.123 vand()**

```
static INLINE CONST vect_t vand (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.288.2.124 vandnot()**

```
static INLINE CONST vect_t vandnot (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.288.3 Field Documentation****12.288.3.1 vect\_size**

```
static const constexpr size_t vect_size = 8 [static], [constexpr], [inherited]
```

**12.288.3.2 alignment**

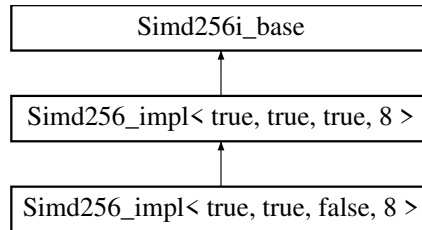
```
static const constexpr size_t alignment = 32 [static], [constexpr], [inherited]
```

The documentation for this struct was generated from the following files:

- [simd256\\_int32.inl](#)
- [simd512\\_int32.inl](#)

## 12.289 Simd256\_impl< true, true, false, 8 > Struct Reference

Inheritance diagram for Simd256\_impl< true, true, false, 8 >:



### Data Structures

- union [Converter](#)

### Public Types

- using [scalar\\_t](#) = uint64\_t
- using [aligned\\_allocator](#) = AlignedAllocator<[scalar\\_t](#), Alignment([alignment](#))>
- using [aligned\\_vector](#) = std::vector<[scalar\\_t](#), [aligned\\_allocator](#)>
- template<class [Field](#) >  
using [is\\_same\\_element](#) = std::is\_same<typename [Field::Element](#), [scalar\\_t](#)>
- using [simdHalf](#) = [Simd128](#)<[scalar\\_t](#)>
- using [vect\\_t](#) = \_\_m256i
- using [half\\_t](#) = \_\_m128i

### Static Public Member Functions

- static const std::string [type\\_string](#) ()
- static [INLINE CONST vect\\_t set1](#) (const [scalar\\_t](#) x)
- static [INLINE CONST vect\\_t set](#) (const [scalar\\_t](#) x0, const [scalar\\_t](#) x1, const [scalar\\_t](#) x2, const [scalar\\_t](#) x3)
- template<class T >  
static [INLINE PURE vect\\_t gather](#) (const [scalar\\_t](#) \*const p, const T \*const idx)
- static [INLINE PURE vect\\_t load](#) (const [scalar\\_t](#) \*const p)
- static [INLINE PURE vect\\_t loadu](#) (const [scalar\\_t](#) \*const p)
- static [INLINE void store](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)
- static [INLINE void storeu](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)
- static [INLINE void stream](#) ([scalar\\_t](#) \*p, const [vect\\_t](#) v)
- template<int s>  
static [INLINE CONST vect\\_t sra](#) (const [vect\\_t](#) a)
- static [INLINE CONST vect\\_t greater](#) ([vect\\_t](#) a, [vect\\_t](#) b)
- static [INLINE CONST vect\\_t lesser](#) ([vect\\_t](#) a, [vect\\_t](#) b)
- static [INLINE CONST vect\\_t greater\\_eq](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t lesser\\_eq](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t mullo](#) ([vect\\_t](#) a, [vect\\_t](#) b)
- static [INLINE CONST vect\\_t mulhi](#) ([vect\\_t](#) a, [vect\\_t](#) b)
- static [INLINE CONST vect\\_t mulx](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t fmaddx](#) (const [vect\\_t](#) c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE vect\\_t fmaddxin](#) ([vect\\_t](#) &c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t fnmaddx](#) (const [vect\\_t](#) c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE vect\\_t fnmaddxin](#) ([vect\\_t](#) &c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t fmsubx](#) (const [vect\\_t](#) c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE vect\\_t fmsubxin](#) ([vect\\_t](#) &c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST scalar\\_t hadd\\_to\\_scal](#) (const [vect\\_t](#) a)

- template<class T >  
static constexpr bool [valid](#) (T \*p)
- template<class T >  
static constexpr bool [compliant](#) (T n)
- static [INLINE CONST vect\\_t set1](#) (const [scalar\\_t](#) x)
- static [INLINE CONST vect\\_t set](#) (const [scalar\\_t](#) x0, const [scalar\\_t](#) x1, const [scalar\\_t](#) x2, const [scalar\\_t](#) x3)
- template<class T >  
static [INLINE PURE vect\\_t gather](#) (const [scalar\\_t](#) \*const p, const T \*const idx)
- template<int idx>  
static [INLINE CONST scalar\\_t get](#) (vect\_t v)
- static [INLINE PURE vect\\_t load](#) (const [scalar\\_t](#) \*const p)
- static [INLINE PURE vect\\_t loadu](#) (const [scalar\\_t](#) \*const p)
- static [INLINE void store](#) ([scalar\\_t](#) \*p, vect\_t v)
- static [INLINE void storeu](#) ([scalar\\_t](#) \*p, vect\_t v)
- static [INLINE void stream](#) ([scalar\\_t](#) \*p, const vect\_t v)
- template<int s>  
static [INLINE CONST vect\\_t sll](#) (const vect\_t a)
- template<int s>  
static [INLINE CONST vect\\_t srl](#) (const vect\_t a)
- template<uint8\_t s>  
static [INLINE CONST vect\\_t shuffle](#) (const vect\_t a)
- static [INLINE CONST vect\\_t unpacklo\\_intrinsic](#) (const vect\_t a, const vect\_t b)
- static [INLINE CONST vect\\_t unpackhi\\_intrinsic](#) (const vect\_t a, const vect\_t b)
- static [INLINE CONST vect\\_t unpacklo](#) (const vect\_t a, const vect\_t b)
- static [INLINE CONST vect\\_t unpackhi](#) (const vect\_t a, const vect\_t b)
- static [INLINE void unpacklohi](#) (vect\_t &lo, vect\_t &hi, const vect\_t a, const vect\_t b)
- static [INLINE CONST vect\\_t pack\\_even](#) (const vect\_t a, const vect\_t b)
- static [INLINE CONST vect\\_t pack\\_odd](#) (const vect\_t a, const vect\_t b)
- static [INLINE void pack](#) (vect\_t &even, vect\_t &odd, const vect\_t a, const vect\_t b)
- static [INLINE void transpose](#) (vect\_t &r0, vect\_t &r1, vect\_t &r2, vect\_t &r3)
- template<uint8\_t s>  
static [INLINE CONST vect\\_t blend](#) (const vect\_t a, const vect\_t b)
- static [INLINE CONST vect\\_t add](#) (const vect\_t a, const vect\_t b)
- static [INLINE vect\\_t addin](#) (vect\_t &a, const vect\_t b)
- static [INLINE CONST vect\\_t sub](#) (const vect\_t a, const vect\_t b)
- static [INLINE vect\\_t subin](#) (vect\_t &a, const vect\_t b)
- static [INLINE CONST vect\\_t mul](#) (const vect\_t a, const vect\_t b)
- static [INLINE CONST vect\\_t fmadd](#) (const vect\_t c, const vect\_t a, const vect\_t b)
- static [INLINE vect\\_t fmaddin](#) (vect\_t &c, const vect\_t a, const vect\_t b)
- static [INLINE CONST vect\\_t fnmadd](#) (const vect\_t c, const vect\_t a, const vect\_t b)
- static [INLINE vect\\_t fnmaddin](#) (vect\_t &c, const vect\_t a, const vect\_t b)
- static [INLINE CONST vect\\_t fmsub](#) (const vect\_t c, const vect\_t a, const vect\_t b)
- static [INLINE vect\\_t fmsubin](#) (vect\_t &c, const vect\_t a, const vect\_t b)
- static [INLINE CONST vect\\_t eq](#) (const vect\_t a, const vect\_t b)
- static [INLINE CONST vect\\_t round](#) (const vect\_t a)
- static [INLINE CONST vect\\_t mask\\_high](#) ()
- static [INLINE CONST vect\\_t mulhi\\_fast](#) (vect\_t x, vect\_t y)
- static [INLINE vect\\_t mod](#) (vect\_t &C, const \_\_m256d &P, const \_\_m256d &INVP, const \_\_m256d &NEGP, const vect\_t &POW50REM, const \_\_m256d &MIN, const \_\_m256d &MAX, \_\_m256d &Q, \_\_m256d &T)
- static [INLINE CONST vect\\_t zero](#) ()

### Static Public Attributes

- static const constexpr size\_t [vect\\_size](#) = 4
- static const constexpr size\_t [alignment](#) = 32

## Static Protected Member Functions

- static `INLINE CONST vect_t signbits` (const `vect_t` x)

## 12.289.1 Member Typedef Documentation

### 12.289.1.1 scalar\_t

```
using scalar_t = uint64_t
```

### 12.289.1.2 aligned\_allocator

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

### 12.289.1.3 aligned\_vector

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

### 12.289.1.4 is\_same\_element

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

### 12.289.1.5 simdHalf

```
using simdHalf = Simd128<scalar_t>
```

### 12.289.1.6 vect\_t

```
using vect_t = __m256i [inherited]
```

### 12.289.1.7 half\_t

```
using half_t = __m128i [inherited]
```

## 12.289.2 Member Function Documentation

### 12.289.2.1 type\_string()

```
static const std::string type_string () [inline], [static]
```

### 12.289.2.2 set1() [1/2]

```
static INLINE CONST vect_t set1 (
    const scalar_t x) [inline], [static]
```

### 12.289.2.3 set() [1/2]

```
static INLINE CONST vect_t set (
    const scalar_t x0,
    const scalar_t x1,
    const scalar_t x2,
    const scalar_t x3) [inline], [static]
```

### 12.289.2.4 gather() [1/2]

```
template<class T >
static INLINE PURE vect_t gather (
    const scalar_t *const p,
    const T *const idx) [inline], [static]
```

**12.289.2.5 load()** [1/2]

```
static INLINE PURE vect_t load (  
    const scalar_t *const p) [inline], [static]
```

**12.289.2.6 loadu()** [1/2]

```
static INLINE PURE vect_t loadu (  
    const scalar_t *const p) [inline], [static]
```

**12.289.2.7 store()** [1/2]

```
static INLINE void store (  
    scalar_t * p,  
    vect_t v) [inline], [static]
```

**12.289.2.8 storeu()** [1/2]

```
static INLINE void storeu (  
    scalar_t * p,  
    vect_t v) [inline], [static]
```

**12.289.2.9 stream()** [1/2]

```
static INLINE void stream (  
    scalar_t * p,  
    const vect_t v) [inline], [static]
```

**12.289.2.10 sra()**

```
template<int s>  
static INLINE CONST vect_t sra (  
    const vect_t a) [inline], [static]
```

**12.289.2.11 greater()**

```
static INLINE CONST vect_t greater (  
    vect_t a,  
    vect_t b) [inline], [static]
```

**12.289.2.12 lesser()**

```
static INLINE CONST vect_t lesser (  
    vect_t a,  
    vect_t b) [inline], [static]
```

**12.289.2.13 greater\_eq()**

```
static INLINE CONST vect_t greater_eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.289.2.14 lesser\_eq()**

```
static INLINE CONST vect_t lesser_eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.289.2.15 mullo()**

```
static INLINE CONST vect_t mullo (  
    vect_t a,  
    vect_t b) [inline], [static]
```

**12.289.2.16 mulhi()**

```
static INLINE CONST vect_t mulhi (  
    vect_t a,  
    vect_t b) [inline], [static]
```

**12.289.2.17 mulx()**

```
static INLINE CONST vect_t mulx (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.289.2.18 fmaddx()**

```
static INLINE CONST vect_t fmaddx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.289.2.19 fmaddxin()**

```
static INLINE vect_t fmaddxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.289.2.20 fnmaddx()**

```
static INLINE CONST vect_t fnmaddx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.289.2.21 fnmaddxin()**

```
static INLINE vect_t fnmaddxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.289.2.22 fmsubx()**

```
static INLINE CONST vect_t fmsubx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.289.2.23 fmsubxin()**

```
static INLINE vect_t fmsubxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.289.2.24 hadd\_to\_scal()**

```
static INLINE CONST scalar_t hadd_to_scal (
    const vect_t a) [inline], [static]
```

**12.289.2.25 valid()**

```
template<class T >
static constexpr bool valid (
    T * p) [inline], [static], [constexpr], [inherited]
```

**12.289.2.26 compliant()**

```
template<class T >
static constexpr bool compliant (
    T n) [inline], [static], [constexpr], [inherited]
```

**12.289.2.27 set1() [2/2]**

```
static INLINE CONST vect_t set1 (
    const scalar_t x) [inline], [static], [inherited]
```

**12.289.2.28 set() [2/2]**

```
static INLINE CONST vect_t set (
    const scalar_t x0,
    const scalar_t x1,
    const scalar_t x2,
    const scalar_t x3) [inline], [static], [inherited]
```

**12.289.2.29 gather() [2/2]**

```
template<class T >
static INLINE PURE vect_t gather (
    const scalar_t *const p,
    const T *const idx) [inline], [static], [inherited]
```

**12.289.2.30 get()**

```
template<int idx>
static INLINE CONST scalar_t get (
    vect_t v) [inline], [static], [inherited]
```

**12.289.2.31 load() [2/2]**

```
static INLINE PURE vect_t load (
    const scalar_t *const p) [inline], [static], [inherited]
```

**12.289.2.32 loadu() [2/2]**

```
static INLINE PURE vect_t loadu (
    const scalar_t *const p) [inline], [static], [inherited]
```

**12.289.2.33 store() [2/2]**

```
static INLINE void store (
    scalar_t * p,
    vect_t v) [inline], [static], [inherited]
```

**12.289.2.34 storeu() [2/2]**

```
static INLINE void storeu (
    scalar_t * p,
    vect_t v) [inline], [static], [inherited]
```

**12.289.2.35 stream() [2/2]**

```
static INLINE void stream (
    scalar_t * p,
    const vect_t v) [inline], [static], [inherited]
```

**12.289.2.36 sll()**

```
template<int s>
static INLINE CONST vect_t sll (
    const vect_t a) [inline], [static], [inherited]
```

**12.289.2.37 srl()**

```
template<int s>
static INLINE CONST vect_t srl (
    const vect_t a) [inline], [static], [inherited]
```

**12.289.2.38 shuffle()**

```
template<uint8_t s>
static INLINE CONST vect_t shuffle (
    const vect_t a) [inline], [static], [inherited]
```

**12.289.2.39 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.289.2.40 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.289.2.41 unpacklo()**

```
static INLINE CONST vect_t unpacklo (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.289.2.42 unpackhi()**

```
static INLINE CONST vect_t unpackhi (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.289.2.43 unpacklohi()**

```
static INLINE void unpacklohi (
    vect_t & lo,
    vect_t & hi,
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```



**12.289.2.44 pack\_even()**

```
static INLINE CONST vect_t pack_even (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.289.2.45 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.289.2.46 pack()**

```
static INLINE void pack (  
    vect_t & even,  
    vect_t & odd,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.289.2.47 transpose()**

```
static INLINE void transpose (  
    vect_t & r0,  
    vect_t & r1,  
    vect_t & r2,  
    vect_t & r3) [inline], [static], [inherited]
```

**12.289.2.48 blend()**

```
template<uint8_t s>  
static INLINE CONST vect_t blend (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.289.2.49 add()**

```
static INLINE CONST vect_t add (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.289.2.50 addin()**

```
static INLINE vect_t addin (  
    vect_t & a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.289.2.51 sub()**

```
static INLINE CONST vect_t sub (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.289.2.52 subin()**

```
static INLINE vect_t subin (  
    vect_t & a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.289.2.53 mul()**

```
static INLINE CONST vect_t mul (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.289.2.54 fmadd()**

```
static INLINE CONST vect_t fmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.289.2.55 fmaddin()**

```
static INLINE vect_t fmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.289.2.56 fnmadd()**

```
static INLINE CONST vect_t fnmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.289.2.57 fnmaddin()**

```
static INLINE vect_t fnmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.289.2.58 fmsub()**

```
static INLINE CONST vect_t fmsub (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.289.2.59 fmsubin()**

```
static INLINE vect_t fmsubin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.289.2.60 eq()**

```
static INLINE CONST vect_t eq (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.289.2.61 round()**

```
static INLINE CONST vect_t round (  
    const vect_t a) [inline], [static], [inherited]
```

**12.289.2.62 mask\_high()**

```
static INLINE CONST vect_t mask_high () [inline], [static], [inherited]
```

**12.289.2.63 mulhi\_fast()**

```
INLINE CONST vect_t mulhi_fast (
    vect_t x,
    vect_t y) [static], [inherited]
```

**12.289.2.64 mod()**

```
INLINE vect_t mod (
    vect_t & C,
    const __m256d & P,
    const __m256d & INVP,
    const __m256d & NEGP,
    const vect_t & POW5OREM,
    const __m256d & MIN,
    const __m256d & MAX,
    __m256d & Q,
    __m256d & T) [static], [inherited]
```

**12.289.2.65 signbits()**

```
static INLINE CONST vect_t signbits (
    const vect_t x) [inline], [static], [protected], [inherited]
```

**12.289.2.66 zero()**

```
static INLINE CONST vect_t zero () [inline], [static], [inherited]
```

**12.289.3 Field Documentation****12.289.3.1 vect\_size**

```
const constexpr size_t vect_size = 4 [static], [constexpr], [inherited]
```

**12.289.3.2 alignment**

```
const constexpr size_t alignment = 32 [static], [constexpr], [inherited]
```

The documentation for this struct was generated from the following file:

- [simd256\\_int64.inl](#)

**12.290 Simd256\_impl< true, true, true, 2 > Struct Reference**

Inheritance diagram for Simd256\_impl< true, true, true, 2 >:



## Data Structures

- union [Converter](#)

## Public Types

- using [vect\\_t](#) = \_\_m256i
- using [half\\_t](#) = \_\_m128i
- using [scalar\\_t](#) = int16\_t
- using [simdHalf](#) = [Simd128](#)<[scalar\\_t](#)>
- using [aligned\\_allocator](#) = [AlignedAllocator](#)<[scalar\\_t](#), [Alignment](#)([alignment](#))>
- using [aligned\\_vector](#) = std::vector<[scalar\\_t](#), [aligned\\_allocator](#)>
- template<class [Field](#) >  
using [is\\_same\\_element](#) = std::is\_same<typename [Field::Element](#), [scalar\\_t](#)>

## Static Public Member Functions

- static const std::string [type\\_string](#) ()
- template<class T >  
static constexpr bool [valid](#) (T \*p)
- template<class T >  
static constexpr bool [compliant](#) (T n)
- static [INLINE CONST vect\\_t set1](#) (const [scalar\\_t](#) x)
- static [INLINE CONST vect\\_t set](#) (const [scalar\\_t](#) x0, const [scalar\\_t](#) x1, const [scalar\\_t](#) x2, const [scalar\\_t](#) x3, const [scalar\\_t](#) x4, const [scalar\\_t](#) x5, const [scalar\\_t](#) x6, const [scalar\\_t](#) x7, const [scalar\\_t](#) x8, const [scalar\\_t](#) x9, const [scalar\\_t](#) x10, const [scalar\\_t](#) x11, const [scalar\\_t](#) x12, const [scalar\\_t](#) x13, const [scalar\\_t](#) x14, const [scalar\\_t](#) x15)
- template<class T >  
static [INLINE PURE vect\\_t gather](#) (const [scalar\\_t](#) \*const p, const T \*const idx)
- static [INLINE PURE vect\\_t load](#) (const [scalar\\_t](#) \*const p)
- static [INLINE PURE vect\\_t loadu](#) (const [scalar\\_t](#) \*const p)
- static [INLINE void store](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)
- static [INLINE void storeu](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)
- static [INLINE void stream](#) ([scalar\\_t](#) \*p, const [vect\\_t](#) v)
- template<int s>  
static [INLINE CONST vect\\_t sll](#) (const [vect\\_t](#) a)
- template<int s>  
static [INLINE CONST vect\\_t srl](#) (const [vect\\_t](#) a)
- template<int s>  
static [INLINE CONST vect\\_t sra](#) (const [vect\\_t](#) a)
- template<uint64\_t s>  
static [INLINE CONST vect\\_t shuffle](#) (const [vect\\_t](#) a)
- static [INLINE CONST vect\\_t unpacklo\\_intrinsic](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpackhi\\_intrinsic](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpacklo](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpackhi](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE void unpacklohi](#) ([vect\\_t](#) &lo, [vect\\_t](#) &hi, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t pack\\_even](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t pack\\_odd](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE void pack](#) ([vect\\_t](#) &even, [vect\\_t](#) &odd, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE void transpose](#) ([vect\\_t](#) &r0, [vect\\_t](#) &r1, [vect\\_t](#) &r2, [vect\\_t](#) &r3, [vect\\_t](#) &r4, [vect\\_t](#) &r5, [vect\\_t](#) &r6, [vect\\_t](#) &r7, [vect\\_t](#) &r8, [vect\\_t](#) &r9, [vect\\_t](#) &r10, [vect\\_t](#) &r11, [vect\\_t](#) &r12, [vect\\_t](#) &r13, [vect\\_t](#) &r14, [vect\\_t](#) &r15)
- template<uint16\_t s, typename std::enable\_if<(s & 0x00ff) == (s > > 8)>::type \* = nullptr>  
static [INLINE vect\\_t blend](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- template<uint16\_t s, typename std::enable\_if<(s & 0x00ff) != (s > > 8)>::type \* = nullptr>  
static [INLINE vect\\_t blend](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t add](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)

- static `INLINE vect_t addin (vect_t &a, const vect_t b)`
- static `INLINE CONST vect_t sub (const vect_t a, const vect_t b)`
- static `INLINE vect_t subin (vect_t &a, const vect_t b)`
- static `INLINE CONST vect_t mullo (const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t mul (const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t mulhi (const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t mulx (vect_t a, vect_t b)`
- static `INLINE CONST vect_t fmadd (const vect_t c, const vect_t a, const vect_t b)`
- static `INLINE vect_t fmaddin (vect_t &c, const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t fmaddx (const vect_t c, const vect_t a, const vect_t b)`
- static `INLINE vect_t fmaddxin (vect_t &c, const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t fnmadd (const vect_t c, const vect_t a, const vect_t b)`
- static `INLINE vect_t fnmaddin (vect_t &c, const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t fnmaddx (const vect_t c, const vect_t a, const vect_t b)`
- static `INLINE vect_t fnmaddxin (vect_t &c, const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t fmsub (const vect_t c, const vect_t a, const vect_t b)`
- static `INLINE vect_t fmsubin (vect_t &c, const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t fmsubx (const vect_t c, const vect_t a, const vect_t b)`
- static `INLINE vect_t fmsubxin (vect_t &c, const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t eq (const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t greater (const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t lesser (const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t greater_eq (const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t lesser_eq (const vect_t a, const vect_t b)`
- static `INLINE CONST scalar_t hadd_to_scal (const vect_t a)`
- static `INLINE CONST vect_t round (const vect_t a)`
- static `INLINE vect_t mod (vect_t &C, const vect_t &P, const vect_t &INVP, const vect_t &NEGP, const vect_t &MIN, const vect_t &MAX, vect_t &Q, vect_t &T)`
- static `INLINE CONST vect_t zero ()`

### Static Public Attributes

- static const constexpr size\_t `vect_size` = 16
- static const constexpr size\_t `alignment` = 32

## 12.290.1 Member Typedef Documentation

### 12.290.1.1 vect\_t

```
using vect_t = __m256i
```

### 12.290.1.2 half\_t

```
using half_t = __m128i
```

### 12.290.1.3 scalar\_t

```
using scalar_t = int16_t
```

### 12.290.1.4 simdHalf

```
using simdHalf = Simd128<scalar_t>
```

### 12.290.1.5 aligned\_allocator

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

### 12.290.1.6 aligned\_vector

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

### 12.290.1.7 is\_same\_element

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

## 12.290.2 Member Function Documentation

### 12.290.2.1 type\_string()

```
static const std::string type_string () [inline], [static]
```

### 12.290.2.2 valid()

```
template<class T >
static constexpr bool valid (
    T * p) [inline], [static], [constexpr]
```

### 12.290.2.3 compliant()

```
template<class T >
static constexpr bool compliant (
    T n) [inline], [static], [constexpr]
```

### 12.290.2.4 set1()

```
static INLINE CONST vect_t set1 (
    const scalar_t x) [inline], [static]
```

### 12.290.2.5 set()

```
static INLINE CONST vect_t set (
    const scalar_t x0,
    const scalar_t x1,
    const scalar_t x2,
    const scalar_t x3,
    const scalar_t x4,
    const scalar_t x5,
    const scalar_t x6,
    const scalar_t x7,
    const scalar_t x8,
    const scalar_t x9,
    const scalar_t x10,
    const scalar_t x11,
    const scalar_t x12,
    const scalar_t x13,
    const scalar_t x14,
    const scalar_t x15) [inline], [static]
```

### 12.290.2.6 gather()

```
template<class T >
static INLINE PURE vect_t gather (
    const scalar_t *const p,
    const T *const idx) [inline], [static]
```

**12.290.2.7 load()**

```
static INLINE PURE vect_t load (  
    const scalar_t *const p) [inline], [static]
```

**12.290.2.8 loadu()**

```
static INLINE PURE vect_t loadu (  
    const scalar_t *const p) [inline], [static]
```

**12.290.2.9 store()**

```
static INLINE void store (  
    scalar_t * p,  
    vect_t v) [inline], [static]
```

**12.290.2.10 storeu()**

```
static INLINE void storeu (  
    scalar_t * p,  
    vect_t v) [inline], [static]
```

**12.290.2.11 stream()**

```
static INLINE void stream (  
    scalar_t * p,  
    const vect_t v) [inline], [static]
```

**12.290.2.12 sll()**

```
template<int s>  
static INLINE CONST vect_t sll (  
    const vect_t a) [inline], [static]
```

**12.290.2.13 srl()**

```
template<int s>  
static INLINE CONST vect_t srl (  
    const vect_t a) [inline], [static]
```

**12.290.2.14 sra()**

```
template<int s>  
static INLINE CONST vect_t sra (  
    const vect_t a) [inline], [static]
```

**12.290.2.15 shuffle()**

```
template<uint64_t s>  
static INLINE CONST vect_t shuffle (  
    const vect_t a) [inline], [static]
```

**12.290.2.16 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.17 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.18 unpacklo()**

```
static INLINE CONST vect_t unpacklo (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.19 unpackhi()**

```
static INLINE CONST vect_t unpackhi (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.20 unpacklohi()**

```
static INLINE void unpacklohi (  
    vect_t & lo,  
    vect_t & hi,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.21 pack\_even()**

```
static INLINE CONST vect_t pack_even (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.22 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.23 pack()**

```
static INLINE void pack (  
    vect_t & even,  
    vect_t & odd,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.24 transpose()**

```
static INLINE void transpose (  
    vect_t & r0,  
    vect_t & r1,  
    vect_t & r2,  
    vect_t & r3,  
    vect_t & r4,  
    vect_t & r5,  
    vect_t & r6,  
    vect_t & r7,  
    vect_t & r8,  
    vect_t & r9,  
    vect_t & r10,
```



```
vect_t & r11,  
vect_t & r12,  
vect_t & r13,  
vect_t & r14,  
vect_t & r15) [inline], [static]
```

#### 12.290.2.25 blend() [1/2]

```
template<uint16_t s, typename std::enable_if<(s &0x00ff)==(s > > 8)>::type * = nullptr>  
static INLINE vect_t blend (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.290.2.26 blend() [2/2]

```
template<uint16_t s, typename std::enable_if<(s &0x00ff) !=(s > > 8)>::type * = nullptr>  
static INLINE vect_t blend (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.290.2.27 add()

```
static INLINE CONST vect_t add (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.290.2.28 addin()

```
static INLINE vect_t addin (  
    vect_t & a,  
    const vect_t b) [inline], [static]
```

#### 12.290.2.29 sub()

```
static INLINE CONST vect_t sub (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.290.2.30 subin()

```
static INLINE vect_t subin (  
    vect_t & a,  
    const vect_t b) [inline], [static]
```

#### 12.290.2.31 mullo()

```
static INLINE CONST vect_t mullo (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.290.2.32 mul()

```
static INLINE CONST vect_t mul (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.33 mulhi()**

```
static INLINE CONST vect_t mulhi (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.34 mulx()**

```
static INLINE CONST vect_t mulx (  
    vect_t a,  
    vect_t b) [inline], [static]
```

**12.290.2.35 fmadd()**

```
static INLINE CONST vect_t fmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.36 fmaddin()**

```
static INLINE vect_t fmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.37 fmaddx()**

```
static INLINE CONST vect_t fmaddx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.38 fmaddxin()**

```
static INLINE vect_t fmaddxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.39 fnmadd()**

```
static INLINE CONST vect_t fnmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.40 fnmaddin()**

```
static INLINE vect_t fnmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.41 fnmaddx()**

```
static INLINE CONST vect_t fnmaddx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.42 fnmaddxin()**

```
static INLINE vect_t fnmaddxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.43 fmsub()**

```
static INLINE CONST vect_t fmsub (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.44 fmsubin()**

```
static INLINE vect_t fmsubin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.45 fmsubx()**

```
static INLINE CONST vect_t fmsubx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.46 fmsubxin()**

```
static INLINE vect_t fmsubxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.47 eq()**

```
static INLINE CONST vect_t eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.48 greater()**

```
static INLINE CONST vect_t greater (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.49 lesser()**

```
static INLINE CONST vect_t lesser (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.50 greater\_eq()**

```
static INLINE CONST vect_t greater_eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.290.2.51 lesser\_eq()**

```
static INLINE CONST vect_t lesser_eq (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.290.2.52 hadd\_to\_scal()**

```
static INLINE CONST scalar_t hadd_to_scal (
    const vect_t a) [inline], [static]
```

**12.290.2.53 round()**

```
static INLINE CONST vect_t round (
    const vect_t a) [inline], [static]
```

**12.290.2.54 mod()**

```
static INLINE vect_t mod (
    vect_t & C,
    const vect_t & P,
    const vect_t & INVP,
    const vect_t & NEGP,
    const vect_t & MIN,
    const vect_t & MAX,
    vect_t & Q,
    vect_t & T) [inline], [static]
```

**12.290.2.55 zero()**

```
static INLINE CONST vect_t zero () [inline], [static], [inherited]
```

**12.290.3 Field Documentation****12.290.3.1 vect\_size**

```
const constexpr size_t vect_size = 16 [static], [constexpr]
```

**12.290.3.2 alignment**

```
const constexpr size_t alignment = 32 [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [simd256\\_int16.inl](#)

**12.291 Simd256\_impl< true, true, true, 4 > Struct Reference**

Inheritance diagram for Simd256\_impl< true, true, true, 4 >:



## Data Structures

- union [Converter](#)

## Public Types

- using [vect\\_t](#) = \_\_m256i
- using [half\\_t](#) = \_\_m128i
- using [scalar\\_t](#) = int32\_t
- using [simdHalf](#) = [Simd128](#)<[scalar\\_t](#)>
- using [aligned\\_allocator](#) = [AlignedAllocator](#)<[scalar\\_t](#), [Alignment](#)([alignment](#))>
- using [aligned\\_vector](#) = std::vector<[scalar\\_t](#), [aligned\\_allocator](#)>
- template<class [Field](#) >  
using [is\\_same\\_element](#) = std::is\_same<typename [Field](#)::Element, [scalar\\_t](#)>
- using [vect\\_t](#) = \_\_m512i
- using [half\\_t](#) = \_\_m256i
- using [simdHalf](#) = [Simd256](#)<[scalar\\_t](#)>

## Public Member Functions

- template<class [Field](#) >  
using [is\\_same\\_element](#)

## Static Public Member Functions

- static const std::string [type\\_string](#) ()
- template<class T >  
static constexpr bool [valid](#) (T \*p)
- template<class T >  
static constexpr bool [compliant](#) (T n)
- static [INLINE CONST](#) [vect\\_t set1](#) (const [scalar\\_t](#) x)
- static [INLINE CONST](#) [vect\\_t set](#) (const [scalar\\_t](#) x0, const [scalar\\_t](#) x1, const [scalar\\_t](#) x2, const [scalar\\_t](#) x3, const [scalar\\_t](#) x4, const [scalar\\_t](#) x5, const [scalar\\_t](#) x6, const [scalar\\_t](#) x7)
- template<class T >  
static [INLINE PURE](#) [vect\\_t gather](#) (const [scalar\\_t](#) \*const p, const T \*const idx)
- static [INLINE PURE](#) [vect\\_t load](#) (const [scalar\\_t](#) \*const p)
- static [INLINE PURE](#) [vect\\_t loadu](#) (const [scalar\\_t](#) \*const p)
- static [INLINE](#) void [store](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)
- static [INLINE](#) void [storeu](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)
- static [INLINE](#) void [stream](#) ([scalar\\_t](#) \*p, const [vect\\_t](#) v)
- template<int s>  
static [INLINE CONST](#) [vect\\_t sll](#) (const [vect\\_t](#) a)
- template<int s>  
static [INLINE CONST](#) [vect\\_t srl](#) (const [vect\\_t](#) a)
- template<int s>  
static [INLINE CONST](#) [vect\\_t sra](#) (const [vect\\_t](#) a)
- template<uint8\_t s>  
static [INLINE CONST](#) [vect\\_t shuffle\\_twice](#) (const [vect\\_t](#) a)
- template<uint32\_t s>  
static [INLINE CONST](#) [vect\\_t shuffle](#) (const [vect\\_t](#) a)
- static [INLINE CONST](#) [vect\\_t unpacklo\\_intrinsic](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST](#) [vect\\_t unpackhi\\_intrinsic](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST](#) [vect\\_t unpacklo](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST](#) [vect\\_t unpackhi](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE](#) void [unpacklohi](#) ([vect\\_t](#) &lo, [vect\\_t](#) &hi, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST](#) [vect\\_t pack\\_even](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST](#) [vect\\_t pack\\_odd](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)

- static `INLINE` void `pack` (`vect_t` &even, `vect_t` &odd, const `vect_t` a, const `vect_t` b)
- static `INLINE` void `transpose` (`vect_t` &r0, `vect_t` &r1, `vect_t` &r2, `vect_t` &r3, `vect_t` &r4, `vect_t` &r5, `vect_t` &r6, `vect_t` &r7)
- template<uint8\_t s>
  - static `INLINE` `CONST` `vect_t` `blend` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `add` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `addin` (`vect_t` &a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `sub` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `subin` (`vect_t` &a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `mullo` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `mul` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `mulhi` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `mulx` (`vect_t` a, `vect_t` b)
- static `INLINE` `CONST` `vect_t` `fmadd` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `fmaddin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `fmaddx` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `fmaddxin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `fnmadd` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `fnmaddin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `fnmaddx` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `fnmaddxin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `fmsub` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `fmsubin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `fmsubx` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `fmsubxin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `greater` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `lesser` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `greater_eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `lesser_eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `scalar_t` `hadd_to_scal` (const `vect_t` a)
- static `INLINE` `CONST` `vect_t` `round` (const `vect_t` a)
- static `INLINE` `vect_t` `mod` (`vect_t` &C, const `vect_t` &P, const `vect_t` &INVP, const `vect_t` &NEGP, const `vect_t` &MIN, const `vect_t` &MAX, `vect_t` &Q, `vect_t` &T)
- static const std::string `type_string` ()
- template<class T>
  - static constexpr bool `valid` (T \*p)
- template<class T>
  - static constexpr bool `compliant` (T n)
- static `INLINE` `CONST` `vect_t` `set1` (const `scalar_t` x)
- static `INLINE` `CONST` `vect_t` `set` (const `scalar_t` x0, const `scalar_t` x1, const `scalar_t` x2, const `scalar_t` x3, const `scalar_t` x4, const `scalar_t` x5, const `scalar_t` x6, const `scalar_t` x7, const `scalar_t` x8, const `scalar_t` x9, const `scalar_t` x10, const `scalar_t` x11, const `scalar_t` x12, const `scalar_t` x13, const `scalar_t` x14, const `scalar_t` x15)
- template<class T>
  - static `INLINE` `PURE` `vect_t` `gather` (const `scalar_t` \*const p, const T \*const idx)
- static `INLINE` `PURE` `vect_t` `load` (const `scalar_t` \*const p)
- static `INLINE` `PURE` `vect_t` `loadu` (const `scalar_t` \*const p)
- static `INLINE` void `store` (`scalar_t` \*p, `vect_t` v)
- static `INLINE` void `storeu` (`scalar_t` \*p, `vect_t` v)
- static `INLINE` void `stream` (`scalar_t` \*p, const `vect_t` v)
- template<int s>
  - static `INLINE` `CONST` `vect_t` `sll` (const `vect_t` a)
- template<int s>
  - static `INLINE` `CONST` `vect_t` `srl` (const `vect_t` a)

- `template<int s>`  
`static INLINE CONST vect_t sra (const vect_t a)`
- `template<uint8_t s>`  
`static INLINE CONST vect_t shuffle_twice (const vect_t a)`
- `template<uint64_t s>`  
`static INLINE CONST vect_t shuffle (const vect_t a)`
- `static INLINE CONST vect_t unpacklo_intrinsic (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t unpackhi_intrinsic (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t unpacklo (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t unpackhi (const vect_t a, const vect_t b)`
- `static INLINE void unpacklohi (vect_t &lo, vect_t &hi, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t pack_even (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t pack_odd (const vect_t a, const vect_t b)`
- `static INLINE void pack (vect_t &even, vect_t &odd, const vect_t a, const vect_t b)`
- `static INLINE void transpose (vect_t &r0, vect_t &r1, vect_t &r2, vect_t &r3, vect_t &r4, vect_t &r5, vect_t &r6, vect_t &r7, vect_t &r8, vect_t &r9, vect_t &r10, vect_t &r11, vect_t &r12, vect_t &r13, vect_t &r14, vect_t &r15)`
- `template<uint16_t s>`  
`static INLINE CONST vect_t blend (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t add (const vect_t a, const vect_t b)`
- `static INLINE vect_t addin (vect_t &a, const vect_t b)`
- `static INLINE CONST vect_t sub (const vect_t a, const vect_t b)`
- `static INLINE vect_t subin (vect_t &a, const vect_t b)`
- `static INLINE CONST vect_t mullo (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mul (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mulhi (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mulx (vect_t a, vect_t b)`
- `static INLINE CONST vect_t fmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmadddin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmadddx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmadddxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fnmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fnmaddin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fnmaddx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fnmaddxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmsub (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmsubin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmsubx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmsubxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t greater (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t lesser (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t greater_eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t lesser_eq (const vect_t a, const vect_t b)`
- `static INLINE CONST scalar_t hadd_to_scal (const vect_t a)`
- `static INLINE CONST vect_t round (const vect_t a)`
- `static INLINE vect_t mod (vect_t &C, const vect_t &P, const vect_t &INVP, const vect_t &NEGP, const vect_t &MIN, const vect_t &MAX, vect_t &Q, vect_t &T)`
- `static INLINE CONST vect_t zero ()`
- `static INLINE CONST vect_t zero ()`
- `static INLINE CONST vect_t vor (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vxor (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vand (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vandnot (const vect_t a, const vect_t b)`

**Static Public Attributes**

- static const constexpr size\_t `vect_size` = 8
- static const constexpr size\_t `alignment` = 32

**12.291.1 Member Typedef Documentation****12.291.1.1 vect\_t [1/2]**

```
using vect_t = __m256i
```

**12.291.1.2 half\_t [1/2]**

```
using half_t = __m128i
```

**12.291.1.3 scalar\_t**

```
typedef int32_t scalar_t = int32_t
```

**12.291.1.4 simdHalf [1/2]**

```
using simdHalf = Simd128<scalar_t>
```

**12.291.1.5 aligned\_allocator**

```
typedef AlignedAllocator< scalar_t, Alignment(alignment)> aligned_allocator = AlignedAllocator<scalar_t,
Alignment(alignment)>
```

**12.291.1.6 aligned\_vector**

```
typedef std::vector< scalar_t, aligned_allocator > aligned_vector = std::vector<scalar_t,
aligned_allocator>
```

**12.291.1.7 is\_same\_element**

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

**12.291.1.8 vect\_t [2/2]**

```
using vect_t = __m512i
```

**12.291.1.9 half\_t [2/2]**

```
using half_t = __m256i
```

**12.291.1.10 simdHalf [2/2]**

```
using simdHalf = Simd256<scalar_t>
```

**12.291.2 Member Function Documentation****12.291.2.1 type\_string() [1/2]**

```
static const std::string type_string () [inline], [static]
```

**12.291.2.2 valid() [1/2]**

```
template<class T >
static constexpr bool valid (
    T * p) [inline], [static], [constexpr]
```



**12.291.2.3 compliant()** [1/2]

```
template<class T >
static constexpr bool compliant (
    T n) [inline], [static], [constexpr]
```

**12.291.2.4 set1()** [1/2]

```
static INLINE CONST vect_t set1 (
    const scalar_t x) [inline], [static]
```

**12.291.2.5 set()** [1/2]

```
static INLINE CONST vect_t set (
    const scalar_t x0,
    const scalar_t x1,
    const scalar_t x2,
    const scalar_t x3,
    const scalar_t x4,
    const scalar_t x5,
    const scalar_t x6,
    const scalar_t x7) [inline], [static]
```

**12.291.2.6 gather()** [1/2]

```
template<class T >
static INLINE PURE vect_t gather (
    const scalar_t *const p,
    const T *const idx) [inline], [static]
```

**12.291.2.7 load()** [1/2]

```
static INLINE PURE vect_t load (
    const scalar_t *const p) [inline], [static]
```

**12.291.2.8 loadu()** [1/2]

```
static INLINE PURE vect_t loadu (
    const scalar_t *const p) [inline], [static]
```

**12.291.2.9 store()** [1/2]

```
static INLINE void store (
    scalar_t * p,
    vect_t v) [inline], [static]
```

**12.291.2.10 storeu()** [1/2]

```
static INLINE void storeu (
    scalar_t * p,
    vect_t v) [inline], [static]
```

**12.291.2.11 stream()** [1/2]

```
static INLINE void stream (
    scalar_t * p,
    const vect_t v) [inline], [static]
```

**12.291.2.12 sll()** [1/2]

```
template<int s>
static INLINE CONST vect_t sll (
    const vect_t a) [inline], [static]
```

**12.291.2.13 srl()** [1/2]

```
template<int s>
static INLINE CONST vect_t srl (
    const vect_t a) [inline], [static]
```

**12.291.2.14 sra()** [1/2]

```
template<int s>
static INLINE CONST vect_t sra (
    const vect_t a) [inline], [static]
```

**12.291.2.15 shuffle\_twice()** [1/2]

```
template<uint8_t s>
static INLINE CONST vect_t shuffle_twice (
    const vect_t a) [inline], [static]
```

**12.291.2.16 shuffle()** [1/2]

```
template<uint32_t s>
static INLINE CONST vect_t shuffle (
    const vect_t a) [inline], [static]
```

**12.291.2.17 unpacklo\_intrinsic()** [1/2]

```
static INLINE CONST vect_t unpacklo_intrinsic (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.291.2.18 unpackhi\_intrinsic()** [1/2]

```
static INLINE CONST vect_t unpackhi_intrinsic (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.291.2.19 unpacklo()** [1/2]

```
static INLINE CONST vect_t unpacklo (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.291.2.20 unpackhi()** [1/2]

```
static INLINE CONST vect_t unpackhi (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.291.2.21 unpacklohi()** [1/2]

```
static INLINE void unpacklohi (
    vect_t & lo,
    vect_t & hi,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.291.2.22 pack\_even()** [1/2]

```
static INLINE CONST vect_t pack_even (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.23 pack\_odd()** [1/2]

```
static INLINE CONST vect_t pack_odd (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.24 pack()** [1/2]

```
static INLINE void pack (  
    vect_t & even,  
    vect_t & odd,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.25 transpose()** [1/2]

```
static INLINE void transpose (  
    vect_t & r0,  
    vect_t & r1,  
    vect_t & r2,  
    vect_t & r3,  
    vect_t & r4,  
    vect_t & r5,  
    vect_t & r6,  
    vect_t & r7) [inline], [static]
```

**12.291.2.26 blend()** [1/2]

```
template<uint8_t s>  
static INLINE CONST vect_t blend (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.27 add()** [1/2]

```
static INLINE CONST vect_t add (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.28 addin()** [1/2]

```
static INLINE vect_t addin (  
    vect_t & a,  
    const vect_t b) [inline], [static]
```

**12.291.2.29 sub()** [1/2]

```
static INLINE CONST vect_t sub (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.30 subin()** [1/2]

```
static INLINE vect_t subin (  
    vect_t & a,  
    const vect_t b) [inline], [static]
```

**12.291.2.31 mullo()** [1/2]

```
static INLINE CONST vect_t mullo (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.32 mul()** [1/2]

```
static INLINE CONST vect_t mul (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.33 mulhi()** [1/2]

```
static INLINE CONST vect_t mulhi (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.34 mulx()** [1/2]

```
static INLINE CONST vect_t mulx (  
    vect_t a,  
    vect_t b) [inline], [static]
```

**12.291.2.35 fmadd()** [1/2]

```
static INLINE CONST vect_t fmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.36 fmaddin()** [1/2]

```
static INLINE vect_t fmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.37 fmaddx()** [1/2]

```
static INLINE CONST vect_t fmaddx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.38 fmaddxin()** [1/2]

```
static INLINE vect_t fmaddxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.39 fnmadd()** [1/2]

```
static INLINE CONST vect_t fnmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.40 fnmaddin()** [1/2]

```
static INLINE vect_t fnmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.41 fnmaddx()** [1/2]

```
static INLINE CONST vect_t fnmaddx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.42 fnmaddxin()** [1/2]

```
static INLINE vect_t fnmaddxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.43 fmsub()** [1/2]

```
static INLINE CONST vect_t fmsub (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.44 fmsubin()** [1/2]

```
static INLINE vect_t fmsubin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.45 fmsubx()** [1/2]

```
static INLINE CONST vect_t fmsubx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.46 fmsubxin()** [1/2]

```
static INLINE vect_t fmsubxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.47 eq()** [1/2]

```
static INLINE CONST vect_t eq (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.291.2.48 greater()** [1/2]

```
static INLINE CONST vect_t greater (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.291.2.49 lesser()** [1/2]

```
static INLINE CONST vect_t lesser (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.291.2.50 greater\_eq()** [1/2]

```
static INLINE CONST vect_t greater_eq (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.291.2.51 lesser\_eq()** [1/2]

```
static INLINE CONST vect_t lesser_eq (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.291.2.52 hadd\_to\_scal()** [1/2]

```
static INLINE CONST scalar_t hadd_to_scal (
    const vect_t a) [inline], [static]
```

**12.291.2.53 round()** [1/2]

```
static INLINE CONST vect_t round (
    const vect_t a) [inline], [static]
```

**12.291.2.54 mod()** [1/2]

```
static INLINE vect_t mod (
    vect_t & C,
    const vect_t & P,
    const vect_t & INVP,
    const vect_t & NEGP,
    const vect_t & MIN,
    const vect_t & MAX,
    vect_t & Q,
    vect_t & T) [inline], [static]
```

**12.291.2.55 type\_string()** [2/2]

```
static const std::string type_string () [inline], [static]
```

**12.291.2.56 valid()** [2/2]

```
template<class T >
static constexpr bool valid (
    T * p) [inline], [static], [constexpr]
```

**12.291.2.57 compliant()** [2/2]

```
template<class T >
static constexpr bool compliant (
    T n) [inline], [static], [constexpr]
```

**12.291.2.58 set1()** [2/2]

```
static INLINE CONST vect_t set1 (
    const scalar_t x) [inline], [static]
```

**12.291.2.59 set()** [2/2]

```
static INLINE CONST vect_t set (
    const scalar_t x0,
    const scalar_t x1,
    const scalar_t x2,
    const scalar_t x3,
    const scalar_t x4,
    const scalar_t x5,
    const scalar_t x6,
    const scalar_t x7,
    const scalar_t x8,
    const scalar_t x9,
    const scalar_t x10,
    const scalar_t x11,
    const scalar_t x12,
    const scalar_t x13,
    const scalar_t x14,
    const scalar_t x15) [inline], [static]
```

**12.291.2.60 gather()** [2/2]

```
template<class T >
static INLINE PURE vect_t gather (
    const scalar_t *const p,
    const T *const idx) [inline], [static]
```

**12.291.2.61 load()** [2/2]

```
static INLINE PURE vect_t load (
    const scalar_t *const p) [inline], [static]
```

**12.291.2.62 loadu()** [2/2]

```
static INLINE PURE vect_t loadu (
    const scalar_t *const p) [inline], [static]
```

**12.291.2.63 store()** [2/2]

```
static INLINE void store (
    scalar_t * p,
    vect_t v) [inline], [static]
```

**12.291.2.64 storeu()** [2/2]

```
static INLINE void storeu (
    scalar_t * p,
    vect_t v) [inline], [static]
```

**12.291.2.65 stream() [2/2]**

```
static INLINE void stream (  
    scalar_t * p,  
    const vect_t v) [inline], [static]
```

**12.291.2.66 sll() [2/2]**

```
template<int s>  
static INLINE CONST vect_t sll (  
    const vect_t a) [inline], [static]
```

**12.291.2.67 srl() [2/2]**

```
template<int s>  
static INLINE CONST vect_t srl (  
    const vect_t a) [inline], [static]
```

**12.291.2.68 sra() [2/2]**

```
template<int s>  
static INLINE CONST vect_t sra (  
    const vect_t a) [inline], [static]
```

**12.291.2.69 shuffle\_twice() [2/2]**

```
template<uint8_t s>  
static INLINE CONST vect_t shuffle_twice (  
    const vect_t a) [inline], [static]
```

**12.291.2.70 shuffle() [2/2]**

```
template<uint64_t s>  
static INLINE CONST vect_t shuffle (  
    const vect_t a) [inline], [static]
```

**12.291.2.71 unpacklo\_intrinsic() [2/2]**

```
static INLINE CONST vect_t unpacklo_intrinsic (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.72 unpackhi\_intrinsic() [2/2]**

```
static INLINE CONST vect_t unpackhi_intrinsic (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.73 unpacklo() [2/2]**

```
static INLINE CONST vect_t unpacklo (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.74 unpackhi() [2/2]**

```
static INLINE CONST vect_t unpackhi (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```



**12.291.2.75 unpacklohi() [2/2]**

```
static INLINE void unpacklohi (  
    vect_t & lo,  
    vect_t & hi,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.76 pack\_even() [2/2]**

```
static INLINE CONST vect_t pack_even (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.77 pack\_odd() [2/2]**

```
static INLINE CONST vect_t pack_odd (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.78 pack() [2/2]**

```
static INLINE void pack (  
    vect_t & even,  
    vect_t & odd,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.79 transpose() [2/2]**

```
static INLINE void transpose (  
    vect_t & r0,  
    vect_t & r1,  
    vect_t & r2,  
    vect_t & r3,  
    vect_t & r4,  
    vect_t & r5,  
    vect_t & r6,  
    vect_t & r7,  
    vect_t & r8,  
    vect_t & r9,  
    vect_t & r10,  
    vect_t & r11,  
    vect_t & r12,  
    vect_t & r13,  
    vect_t & r14,  
    vect_t & r15) [inline], [static]
```

**12.291.2.80 blend() [2/2]**

```
template<uint16_t s>  
static INLINE CONST vect_t blend (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.81 add() [2/2]**

```
static INLINE CONST vect_t add (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.82 addin()** [2/2]

```
static INLINE vect_t addin (  
    vect_t & a,  
    const vect_t b) [inline], [static]
```

**12.291.2.83 sub()** [2/2]

```
static INLINE CONST vect_t sub (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.84 subin()** [2/2]

```
static INLINE vect_t subin (  
    vect_t & a,  
    const vect_t b) [inline], [static]
```

**12.291.2.85 mullo()** [2/2]

```
static INLINE CONST vect_t mullo (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.86 mul()** [2/2]

```
static INLINE CONST vect_t mul (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.87 mulhi()** [2/2]

```
static INLINE CONST vect_t mulhi (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.88 mulx()** [2/2]

```
static INLINE CONST vect_t mulx (  
    vect_t a,  
    vect_t b) [inline], [static]
```

**12.291.2.89 fmadd()** [2/2]

```
static INLINE CONST vect_t fmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.90 fmaddin()** [2/2]

```
static INLINE vect_t fmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.91 fmaddx()** [2/2]

```
static INLINE CONST vect_t fmaddx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.92 fmaddxin()** [2/2]

```
static INLINE vect_t fmaddxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.93 fnmadd()** [2/2]

```
static INLINE CONST vect_t fnmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.94 fnmaddin()** [2/2]

```
static INLINE vect_t fnmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.95 fnmaddx()** [2/2]

```
static INLINE CONST vect_t fnmaddx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.96 fnmaddxin()** [2/2]

```
static INLINE vect_t fnmaddxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.97 fmsub()** [2/2]

```
static INLINE CONST vect_t fmsub (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.98 fmsubin()** [2/2]

```
static INLINE vect_t fmsubin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.291.2.99 fmsubx()** [2/2]

```
static INLINE CONST vect_t fmsubx (  
    const vect_t c,
```

```
const vect_t a,
const vect_t b) [inline], [static]
```

#### 12.291.2.100 fmsubxin() [2/2]

```
static INLINE vect_t fmsubxin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

#### 12.291.2.101 eq() [2/2]

```
static INLINE CONST vect_t eq (
    const vect_t a,
    const vect_t b) [inline], [static]
```

#### 12.291.2.102 greater() [2/2]

```
static INLINE CONST vect_t greater (
    const vect_t a,
    const vect_t b) [inline], [static]
```

#### 12.291.2.103 lesser() [2/2]

```
static INLINE CONST vect_t lesser (
    const vect_t a,
    const vect_t b) [inline], [static]
```

#### 12.291.2.104 greater\_eq() [2/2]

```
static INLINE CONST vect_t greater_eq (
    const vect_t a,
    const vect_t b) [inline], [static]
```

#### 12.291.2.105 lesser\_eq() [2/2]

```
static INLINE CONST vect_t lesser_eq (
    const vect_t a,
    const vect_t b) [inline], [static]
```

#### 12.291.2.106 hadd\_to\_scal() [2/2]

```
static INLINE CONST scalar_t hadd_to_scal (
    const vect_t a) [inline], [static]
```

#### 12.291.2.107 round() [2/2]

```
static INLINE CONST vect_t round (
    const vect_t a) [inline], [static]
```

#### 12.291.2.108 mod() [2/2]

```
static INLINE vect_t mod (
    vect_t & C,
    const vect_t & P,
    const vect_t & INVP,
    const vect_t & NEGP,
    const vect_t & MIN,
    const vect_t & MAX,
```

```

    vect_t & Q,
    vect_t & T) [inline], [static]

```

#### 12.291.2.109 is\_same\_element()

```

template<class Field >
typedef std::is_same<typename Field::Element, scalar_t> is_same_element

```

#### 12.291.2.110 zero() [1/2]

```

static INLINE CONST vect_t zero () [inline], [static], [inherited]

```

#### 12.291.2.111 zero() [2/2]

```

static INLINE CONST vect_t zero () [inline], [static], [inherited]

```

#### 12.291.2.112 vor()

```

static INLINE CONST vect_t vor (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]

```

#### 12.291.2.113 vxor()

```

static INLINE CONST vect_t vxor (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]

```

#### 12.291.2.114 vand()

```

static INLINE CONST vect_t vand (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]

```

#### 12.291.2.115 vandnot()

```

static INLINE CONST vect_t vandnot (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]

```

### 12.291.3 Field Documentation

#### 12.291.3.1 vect\_size

```

static const constexpr size_t vect_size = 8 [static], [constexpr]

```

#### 12.291.3.2 alignment

```

static const constexpr size_t alignment = 32 [static], [constexpr]

```

The documentation for this struct was generated from the following files:

- [simd256\\_int32.inl](#)
- [simd512\\_int32.inl](#)

## 12.292 Simd256\_impl< true, true, true, 8 > Struct Reference

Inheritance diagram for Simd256\_impl< true, true, true, 8 >:



## Data Structures

- union [Converter](#)

## Public Types

- using [vect\\_t](#) = \_\_m256i
- using [half\\_t](#) = \_\_m128i
- using [scalar\\_t](#) = int64\_t
- using [simdHalf](#) = [Simd128](#)<[scalar\\_t](#)>
- using [aligned\\_allocator](#) = [AlignedAllocator](#)<[scalar\\_t](#), [Alignment](#)([alignment](#))>
- using [aligned\\_vector](#) = std::vector<[scalar\\_t](#), [aligned\\_allocator](#)>
- template<class [Field](#) >  
using [is\\_same\\_element](#) = std::is\_same<typename [Field](#)::[Element](#), [scalar\\_t](#)>

## Static Public Member Functions

- static const std::string [type\\_string](#) ()
- template<class T >  
static constexpr bool [valid](#) (T \*p)
- template<class T >  
static constexpr bool [compliant](#) (T n)
- static [INLINE CONST vect\\_t set1](#) (const [scalar\\_t](#) x)
- static [INLINE CONST vect\\_t set](#) (const [scalar\\_t](#) x0, const [scalar\\_t](#) x1, const [scalar\\_t](#) x2, const [scalar\\_t](#) x3)
- template<class T >  
static [INLINE PURE vect\\_t gather](#) (const [scalar\\_t](#) \*const p, const T \*const idx)
- template<int idx>  
static [INLINE CONST scalar\\_t get](#) ([vect\\_t](#) v)
- static [INLINE PURE vect\\_t load](#) (const [scalar\\_t](#) \*const p)
- static [INLINE PURE vect\\_t loadu](#) (const [scalar\\_t](#) \*const p)
- static [INLINE void store](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)
- static [INLINE void storeu](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)
- static [INLINE void stream](#) ([scalar\\_t](#) \*p, const [vect\\_t](#) v)
- template<int s>  
static [INLINE CONST vect\\_t sll](#) (const [vect\\_t](#) a)
- template<int s>  
static [INLINE CONST vect\\_t srl](#) (const [vect\\_t](#) a)
- template<int s>  
static [INLINE CONST vect\\_t sra](#) (const [vect\\_t](#) a)
- template<uint8\_t s>  
static [INLINE CONST vect\\_t shuffle](#) (const [vect\\_t](#) a)
- static [INLINE CONST vect\\_t unpacklo\\_intrinsic](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpackhi\\_intrinsic](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpacklo](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpackhi](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE void unpacklohi](#) ([vect\\_t](#) &lo, [vect\\_t](#) &hi, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t pack\\_even](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t pack\\_odd](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)

- static `INLINE` void `pack` (`vect_t` &even, `vect_t` &odd, const `vect_t` a, const `vect_t` b)
- static `INLINE` void `transpose` (`vect_t` &r0, `vect_t` &r1, `vect_t` &r2, `vect_t` &r3)
- template<uint8\_t s>
  - static `INLINE` `CONST` `vect_t` `blend` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `add` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `addin` (`vect_t` &a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `sub` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `subin` (`vect_t` &a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `mullo` (`vect_t` a, `vect_t` b)
- static `INLINE` `CONST` `vect_t` `mul` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `mulhi` (`vect_t` a, `vect_t` b)
- static `INLINE` `CONST` `vect_t` `mulx` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `fmadd` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `fmaddin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `fmaddx` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `fmaddxin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `fnmadd` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `fnmaddin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `fnmaddx` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `fnmaddxin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `fmsub` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `fmsubin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `fmsubx` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `vect_t` `fmsubxin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `greater` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `lesser` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `greater_eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `lesser_eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `scalar_t` `hadd_to_scal` (const `vect_t` a)
- static `INLINE` `CONST` `vect_t` `round` (const `vect_t` a)
- static `INLINE` `CONST` `vect_t` `mask_high` ()
- static `INLINE` `CONST` `vect_t` `mulhi_fast` (`vect_t` x, `vect_t` y)
- static `INLINE` `vect_t` `mod` (`vect_t` &C, const \_\_m256d &P, const \_\_m256d &INVP, const \_\_m256d &NEGP, const `vect_t` &POW50REM, const \_\_m256d &MIN, const \_\_m256d &MAX, \_\_m256d &Q, \_\_m256d &T)
- static `INLINE` `CONST` `vect_t` `zero` ()

### Static Public Attributes

- static const constexpr size\_t `vect_size` = 4
- static const constexpr size\_t `alignment` = 32

### Static Protected Member Functions

- static `INLINE` `CONST` `vect_t` `signbits` (const `vect_t` x)

## 12.292.1 Member Typedef Documentation

### 12.292.1.1 vect\_t

using `vect_t` = \_\_m256i

### 12.292.1.2 half\_t

using `half_t` = \_\_m128i

**12.292.1.3 scalar\_t**

```
using scalar_t = int64_t
```

**12.292.1.4 simdHalf**

```
using simdHalf = Simd128<scalar_t>
```

**12.292.1.5 aligned\_allocator**

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

**12.292.1.6 aligned\_vector**

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

**12.292.1.7 is\_same\_element**

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

**12.292.2 Member Function Documentation****12.292.2.1 type\_string()**

```
static const std::string type_string () [inline], [static]
```

**12.292.2.2 valid()**

```
template<class T >
static constexpr bool valid (
    T * p) [inline], [static], [constexpr]
```

**12.292.2.3 compliant()**

```
template<class T >
static constexpr bool compliant (
    T n) [inline], [static], [constexpr]
```

**12.292.2.4 set1()**

```
static INLINE CONST vect_t set1 (
    const scalar_t x) [inline], [static]
```

**12.292.2.5 set()**

```
static INLINE CONST vect_t set (
    const scalar_t x0,
    const scalar_t x1,
    const scalar_t x2,
    const scalar_t x3) [inline], [static]
```

**12.292.2.6 gather()**

```
template<class T >
static INLINE PURE vect_t gather (
    const scalar_t *const p,
    const T *const idx) [inline], [static]
```



**12.292.2.7 get()**

```
template<int idx>
static INLINE CONST scalar_t get (
    vect_t v) [inline], [static]
```

**12.292.2.8 load()**

```
static INLINE PURE vect_t load (
    const scalar_t *const p) [inline], [static]
```

**12.292.2.9 loadu()**

```
static INLINE PURE vect_t loadu (
    const scalar_t *const p) [inline], [static]
```

**12.292.2.10 store()**

```
static INLINE void store (
    scalar_t * p,
    vect_t v) [inline], [static]
```

**12.292.2.11 storeu()**

```
static INLINE void storeu (
    scalar_t * p,
    vect_t v) [inline], [static]
```

**12.292.2.12 stream()**

```
static INLINE void stream (
    scalar_t * p,
    const vect_t v) [inline], [static]
```

**12.292.2.13 sll()**

```
template<int s>
static INLINE CONST vect_t sll (
    const vect_t a) [inline], [static]
```

**12.292.2.14 srl()**

```
template<int s>
static INLINE CONST vect_t srl (
    const vect_t a) [inline], [static]
```

**12.292.2.15 sra()**

```
template<int s>
static INLINE CONST vect_t sra (
    const vect_t a) [inline], [static]
```

**12.292.2.16 shuffle()**

```
template<uint8_t s>
static INLINE CONST vect_t shuffle (
    const vect_t a) [inline], [static]
```

**12.292.2.17 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.292.2.18 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.292.2.19 unpacklo()**

```
static INLINE CONST vect_t unpacklo (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.292.2.20 unpackhi()**

```
static INLINE CONST vect_t unpackhi (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.292.2.21 unpacklohi()**

```
static INLINE void unpacklohi (  
    vect_t & lo,  
    vect_t & hi,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.292.2.22 pack\_even()**

```
static INLINE CONST vect_t pack_even (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.292.2.23 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.292.2.24 pack()**

```
static INLINE void pack (  
    vect_t & even,  
    vect_t & odd,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.292.2.25 transpose()**

```
static INLINE void transpose (  
    vect_t & r0,  
    vect_t & r1,  
    vect_t & r2,  
    vect_t & r3) [inline], [static]
```

**12.292.2.26 blend()**

```
template<uint8_t s>
static INLINE CONST vect_t blend (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.292.2.27 add()**

```
static INLINE CONST vect_t add (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.292.2.28 addin()**

```
static INLINE vect_t addin (
    vect_t & a,
    const vect_t b) [inline], [static]
```

**12.292.2.29 sub()**

```
static INLINE CONST vect_t sub (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.292.2.30 subin()**

```
static INLINE vect_t subin (
    vect_t & a,
    const vect_t b) [inline], [static]
```

**12.292.2.31 mullo()**

```
static INLINE CONST vect_t mullo (
    vect_t a,
    vect_t b) [inline], [static]
```

**12.292.2.32 mul()**

```
static INLINE CONST vect_t mul (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.292.2.33 mulhi()**

```
static INLINE CONST vect_t mulhi (
    vect_t a,
    vect_t b) [inline], [static]
```

**12.292.2.34 mulx()**

```
static INLINE CONST vect_t mulx (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.292.2.35 fmadd()**

```
static INLINE CONST vect_t fmadd (
    const vect_t c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.292.2.36 fmaddin()**

```
static INLINE vect_t fmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.292.2.37 fmaddx()**

```
static INLINE CONST vect_t fmaddx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.292.2.38 fmaddxin()**

```
static INLINE vect_t fmaddxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.292.2.39 fnmadd()**

```
static INLINE CONST vect_t fnmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.292.2.40 fnmaddin()**

```
static INLINE vect_t fnmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.292.2.41 fnmaddx()**

```
static INLINE CONST vect_t fnmaddx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.292.2.42 fnmaddxin()**

```
static INLINE vect_t fnmaddxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.292.2.43 fmsub()**

```
static INLINE CONST vect_t fmsub (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.292.2.44 fmsubin()**

```
static INLINE vect_t fmsubin (  
    vect_t & c,
```

```
const vect_t a,  
const vect_t b) [inline], [static]
```

#### 12.292.2.45 fmsubx()

```
static INLINE CONST vect_t fmsubx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.292.2.46 fmsubxin()

```
static INLINE vect_t fmsubxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.292.2.47 eq()

```
static INLINE CONST vect_t eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.292.2.48 greater()

```
static INLINE CONST vect_t greater (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.292.2.49 lesser()

```
static INLINE CONST vect_t lesser (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.292.2.50 greater\_eq()

```
static INLINE CONST vect_t greater_eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.292.2.51 lesser\_eq()

```
static INLINE CONST vect_t lesser_eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.292.2.52 hadd\_to\_scal()

```
static INLINE CONST scalar_t hadd_to_scal (  
    const vect_t a) [inline], [static]
```

#### 12.292.2.53 round()

```
static INLINE CONST vect_t round (  
    const vect_t a) [inline], [static]
```

#### 12.292.2.54 mask\_high()

```
static INLINE CONST vect_t mask_high () [inline], [static]
```

**12.292.2.55 mulhi\_fast()**

```

INLINE CONST vect_t mulhi_fast (
    vect_t x,
    vect_t y) [static]

```

**12.292.2.56 mod()**

```

INLINE vect_t mod (
    vect_t & C,
    const __m256d & P,
    const __m256d & INVP,
    const __m256d & NEGP,
    const vect_t & POW50REM,
    const __m256d & MIN,
    const __m256d & MAX,
    __m256d & Q,
    __m256d & T) [static]

```

**12.292.2.57 signbits()**

```

static INLINE CONST vect_t signbits (
    const vect_t x) [inline], [static], [protected]

```

**12.292.2.58 zero()**

```

static INLINE CONST vect_t zero () [inline], [static], [inherited]

```

**12.292.3 Field Documentation****12.292.3.1 vect\_size**

```

const constexpr size_t vect_size = 4 [static], [constexpr]

```

**12.292.3.2 alignment**

```

const constexpr size_t alignment = 32 [static], [constexpr]

```

The documentation for this struct was generated from the following file:

- [simd256\\_int64.inl](#)

**12.293 Simd256fp\_base Struct Reference**

Inheritance diagram for Simd256fp\_base:



The documentation for this struct was generated from the following file:

- [simd256.inl](#)

## 12.294 Simd256i\_base Struct Reference

Inheritance diagram for Simd256i\_base:



### Public Types

- using [vect\\_t](#) = \_\_m256i

### Static Public Member Functions

- static [INLINE CONST vect\\_t zero](#) ()

### 12.294.1 Member Typedef Documentation

#### 12.294.1.1 vect\_t

using [vect\\_t](#) = \_\_m256i

### 12.294.2 Member Function Documentation

#### 12.294.2.1 zero()

static [INLINE CONST vect\\_t zero](#) () [inline], [static]

The documentation for this struct was generated from the following file:

- [simd256.inl](#)

## 12.295 Simd512\_impl< ArithType, Int, Signed, Size > Struct Template Reference

The documentation for this struct was generated from the following file:

- [simd512.inl](#)

## 12.296 Simd512\_impl< true, false, true, 4 > Struct Reference

The documentation for this struct was generated from the following file:

- [simd512\\_float.inl](#)

## 12.297 Simd512\_impl< true, false, true, 8 > Struct Reference

### Public Types

- using [vect\\_t](#) = \_\_m512d
- using [scalar\\_t](#) = double
- using [aligned\\_allocator](#) = AlignedAllocator<[scalar\\_t](#), Alignment([alignment](#))>
- using [aligned\\_vector](#) = std::vector<[scalar\\_t](#), [aligned\\_allocator](#)>
- template<class [Field](#) >
  - using [is\\_same\\_element](#) = std::is\_same<typename [Field::Element](#), [scalar\\_t](#)>

## Static Public Member Functions

- static const std::string [type\\_string](#) ()
- template<class T >  
static constexpr bool [valid](#) (T \*p)
- template<class T >  
static constexpr bool [compliant](#) (T n)
- static [INLINE CONST vect\\_t zero](#) ()
- static [INLINE CONST vect\\_t set1](#) (const [scalar\\_t](#) x)
- static [INLINE CONST vect\\_t set](#) (const [scalar\\_t](#) x1, const [scalar\\_t](#) x2, const [scalar\\_t](#) x3, const [scalar\\_t](#) x4, const [scalar\\_t](#) x5, const [scalar\\_t](#) x6, const [scalar\\_t](#) x7, const [scalar\\_t](#) x8)
- template<class T >  
static [INLINE PURE vect\\_t gather](#) (const [scalar\\_t](#) \*const p, const T \*const idx)
- static [INLINE PURE vect\\_t load](#) (const [scalar\\_t](#) \*const p)
- static [INLINE PURE vect\\_t loadu](#) (const [scalar\\_t](#) \*const p)
- static [INLINE void store](#) (const [scalar\\_t](#) \*p, const [vect\\_t](#) v)
- static [INLINE void storeu](#) (const [scalar\\_t](#) \*p, const [vect\\_t](#) v)
- static [INLINE void stream](#) (const [scalar\\_t](#) \*p, const [vect\\_t](#) v)
- template<uint8\_t s>  
static [INLINE CONST vect\\_t shuffle](#) (const [vect\\_t](#) a)
- static [INLINE CONST vect\\_t unpacklo\\_intrinsic](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpackhi\\_intrinsic](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpacklo](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpackhi](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE void unpacklohi](#) ([vect\\_t](#) &lo, [vect\\_t](#) &hi, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t pack\\_even](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t pack\\_odd](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE void pack](#) ([vect\\_t](#) &even, [vect\\_t](#) &odd, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE void transpose](#) ([vect\\_t](#) &r0, [vect\\_t](#) &r1, [vect\\_t](#) &r2, [vect\\_t](#) &r3, [vect\\_t](#) &r4, [vect\\_t](#) &r5, [vect\\_t](#) &r6, [vect\\_t](#) &r7)
- template<uint8\_t s>  
static [INLINE CONST vect\\_t blend](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t blendv](#) (const [vect\\_t](#) a, const [vect\\_t](#) b, const [vect\\_t](#) mask)
- static [INLINE CONST vect\\_t add](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE vect\\_t addin](#) ([vect\\_t](#) &a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t sub](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t subin](#) ([vect\\_t](#) &a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t mul](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t mulin](#) ([vect\\_t](#) &a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t div](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t fmad](#) (const [vect\\_t](#) c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t fmadin](#) ([vect\\_t](#) &c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t fnmad](#) (const [vect\\_t](#) c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t fnmadin](#) ([vect\\_t](#) &c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t fmsub](#) (const [vect\\_t](#) c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t fmsubin](#) ([vect\\_t](#) &c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t eq](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t lesser](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t lesser\\_eq](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t greater](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t greater\\_eq](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t floor](#) (const [vect\\_t](#) a)
- static [INLINE CONST vect\\_t ceil](#) (const [vect\\_t](#) a)
- static [INLINE CONST vect\\_t round](#) (const [vect\\_t](#) a)
- static [INLINE CONST vect\\_t hadd](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST scalar\\_t hadd\\_to\\_scal](#) (const [vect\\_t](#) a)



### Static Public Attributes

- static const constexpr size\_t `vect_size` = 8
- static const constexpr size\_t `alignment` = 64

## 12.297.1 Member Typedef Documentation

### 12.297.1.1 vect\_t

```
using vect_t = __m512d
```

### 12.297.1.2 scalar\_t

```
using scalar_t = double
```

### 12.297.1.3 aligned\_allocator

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

### 12.297.1.4 aligned\_vector

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

### 12.297.1.5 is\_same\_element

```
template<class Field >  
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

## 12.297.2 Member Function Documentation

### 12.297.2.1 type\_string()

```
static const std::string type_string () [inline], [static]
```

### 12.297.2.2 valid()

```
template<class T >  
static constexpr bool valid (  
    T * p) [inline], [static], [constexpr]
```

### 12.297.2.3 compliant()

```
template<class T >  
static constexpr bool compliant (  
    T n) [inline], [static], [constexpr]
```

### 12.297.2.4 zero()

```
static INLINE CONST vect_t zero () [inline], [static]
```

### 12.297.2.5 set1()

```
static INLINE CONST vect_t set1 (  
    const scalar_t x) [inline], [static]
```

**12.297.2.6 set()**

```
static INLINE CONST vect_t set (
    const scalar_t x1,
    const scalar_t x2,
    const scalar_t x3,
    const scalar_t x4,
    const scalar_t x5,
    const scalar_t x6,
    const scalar_t x7,
    const scalar_t x8) [inline], [static]
```

**12.297.2.7 gather()**

```
template<class T >
static INLINE PURE vect_t gather (
    const scalar_t *const p,
    const T *const idx) [inline], [static]
```

**12.297.2.8 load()**

```
static INLINE PURE vect_t load (
    const scalar_t *const p) [inline], [static]
```

**12.297.2.9 loadu()**

```
static INLINE PURE vect_t loadu (
    const scalar_t *const p) [inline], [static]
```

**12.297.2.10 store()**

```
static INLINE void store (
    const scalar_t * p,
    const vect_t v) [inline], [static]
```

**12.297.2.11 storeu()**

```
static INLINE void storeu (
    const scalar_t * p,
    const vect_t v) [inline], [static]
```

**12.297.2.12 stream()**

```
static INLINE void stream (
    const scalar_t * p,
    const vect_t v) [inline], [static]
```

**12.297.2.13 shuffle()**

```
template<uint8_t s>
static INLINE CONST vect_t shuffle (
    const vect_t a) [inline], [static]
```

**12.297.2.14 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.297.2.15 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.297.2.16 unpacklo()**

```
static INLINE CONST vect_t unpacklo (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.297.2.17 unpackhi()**

```
static INLINE CONST vect_t unpackhi (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.297.2.18 unpacklohi()**

```
static INLINE void unpacklohi (  
    vect_t & lo,  
    vect_t & hi,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.297.2.19 pack\_even()**

```
static INLINE CONST vect_t pack_even (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.297.2.20 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.297.2.21 pack()**

```
static INLINE void pack (  
    vect_t & even,  
    vect_t & odd,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.297.2.22 transpose()**

```
static INLINE void transpose (  
    vect_t & r0,  
    vect_t & r1,  
    vect_t & r2,  
    vect_t & r3,  
    vect_t & r4,  
    vect_t & r5,  
    vect_t & r6,  
    vect_t & r7) [inline], [static]
```

**12.297.2.23 blend()**

```
template<uint8_t s>
static INLINE CONST vect_t blend (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.297.2.24 blendv()**

```
static INLINE CONST vect_t blendv (
    const vect_t a,
    const vect_t b,
    const vect_t mask) [inline], [static]
```

**12.297.2.25 add()**

```
static INLINE CONST vect_t add (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.297.2.26 addin()**

```
static INLINE vect_t addin (
    vect_t & a,
    const vect_t b) [inline], [static]
```

**12.297.2.27 sub()**

```
static INLINE CONST vect_t sub (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.297.2.28 subin()**

```
static INLINE CONST vect_t subin (
    vect_t & a,
    const vect_t b) [inline], [static]
```

**12.297.2.29 mul()**

```
static INLINE CONST vect_t mul (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.297.2.30 mulin()**

```
static INLINE CONST vect_t mulin (
    vect_t & a,
    const vect_t b) [inline], [static]
```

**12.297.2.31 div()**

```
static INLINE CONST vect_t div (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.297.2.32 fmadd()**

```
static INLINE CONST vect_t fmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.297.2.33 fmaddin()**

```
static INLINE CONST vect_t fmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.297.2.34 fnmadd()**

```
static INLINE CONST vect_t fnmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.297.2.35 fnmaddin()**

```
static INLINE CONST vect_t fnmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.297.2.36 fmsub()**

```
static INLINE CONST vect_t fmsub (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.297.2.37 fmsubin()**

```
static INLINE CONST vect_t fmsubin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.297.2.38 eq()**

```
static INLINE CONST vect_t eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.297.2.39 lesser()**

```
static INLINE CONST vect_t lesser (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.297.2.40 lesser\_eq()**

```
static INLINE CONST vect_t lesser_eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.297.2.41 greater()**

```
static INLINE CONST vect_t greater (
    const vect_t a,
    const vect_t b)  [inline], [static]
```

**12.297.2.42 greater\_eq()**

```
static INLINE CONST vect_t greater_eq (
    const vect_t a,
    const vect_t b)  [inline], [static]
```

**12.297.2.43 floor()**

```
static INLINE CONST vect_t floor (
    const vect_t a)  [inline], [static]
```

**12.297.2.44 ceil()**

```
static INLINE CONST vect_t ceil (
    const vect_t a)  [inline], [static]
```

**12.297.2.45 round()**

```
static INLINE CONST vect_t round (
    const vect_t a)  [inline], [static]
```

**12.297.2.46 hadd()**

```
static INLINE CONST vect_t hadd (
    const vect_t a,
    const vect_t b)  [inline], [static]
```

**12.297.2.47 hadd\_to\_scal()**

```
static INLINE CONST scalar_t hadd_to_scal (
    const vect_t a)  [inline], [static]
```

**12.297.3 Field Documentation****12.297.3.1 vect\_size**

```
const constexpr size_t vect_size = 8  [static], [constexpr]
```

**12.297.3.2 alignment**

```
const constexpr size_t alignment = 64  [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [simd512\\_double.inl](#)

**12.298 Simd512\_impl< true, true, false, 8 > Struct Reference**

Inheritance diagram for Simd512\_impl< true, true, false, 8 >:



## Data Structures

- union [Converter](#)

## Public Types

- using [scalar\\_t](#) = uint64\_t
- using [aligned\\_allocator](#) = AlignedAllocator<[scalar\\_t](#), Alignment([alignment](#))>
- using [aligned\\_vector](#) = std::vector<[scalar\\_t](#), [aligned\\_allocator](#)>
- template<class [Field](#) >  
using [is\\_same\\_element](#) = std::is\_same<typename [Field::Element](#), [scalar\\_t](#)>
- using [simdHalf](#) = Simd256<[scalar\\_t](#)>
- using [vect\\_t](#) = \_\_m512i
- using [half\\_t](#) = \_\_m256i

## Static Public Member Functions

- static const std::string [type\\_string](#) ()
- static [INLINE CONST vect\\_t set1](#) (const [scalar\\_t](#) x)
- static [INLINE CONST vect\\_t set](#) (const [scalar\\_t](#) x0, const [scalar\\_t](#) x1, const [scalar\\_t](#) x2, const [scalar\\_t](#) x3, const [scalar\\_t](#) x4, const [scalar\\_t](#) x5, const [scalar\\_t](#) x6, const [scalar\\_t](#) x7)
- template<class T >  
static [INLINE PURE vect\\_t gather](#) (const [scalar\\_t](#) \*const p, const T \*const idx)
- static [INLINE PURE vect\\_t load](#) (const [scalar\\_t](#) \*const p)
- static [INLINE PURE vect\\_t loadu](#) (const [scalar\\_t](#) \*const p)
- static [INLINE](#) void [store](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)
- template<uint8\_t k>  
static [INLINE](#) void [maskstore](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)
- static [INLINE](#) void [storeu](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)
- static [INLINE](#) void [stream](#) ([scalar\\_t](#) \*p, const [vect\\_t](#) v)
- template<int s>  
static [INLINE CONST vect\\_t sra](#) (const [vect\\_t](#) a)
- static [INLINE CONST vect\\_t greater](#) ([vect\\_t](#) a, [vect\\_t](#) b)
- static [INLINE CONST vect\\_t lesser](#) ([vect\\_t](#) a, [vect\\_t](#) b)
- static [INLINE CONST vect\\_t greater\\_eq](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t lesser\\_eq](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t mullo](#) ([vect\\_t](#) a, [vect\\_t](#) b)
- static [INLINE CONST vect\\_t mulhi](#) ([vect\\_t](#) a, [vect\\_t](#) b)
- static [INLINE CONST vect\\_t mulx](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t fmaddx](#) (const [vect\\_t](#) c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE vect\\_t fmaddxin](#) ([vect\\_t](#) &c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t fnmaddx](#) (const [vect\\_t](#) c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE vect\\_t fnmaddxin](#) ([vect\\_t](#) &c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t fmsubx](#) (const [vect\\_t](#) c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE vect\\_t fmsubxin](#) ([vect\\_t](#) &c, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST scalar\\_t hadd\\_to\\_scal](#) (const [vect\\_t](#) a)
- template<class T >  
static constexpr bool [valid](#) (T \*p)

- `template<class T >`  
`static constexpr bool compliant (T n)`
- `static INLINE CONST vect_t set1 (const scalar_t x)`
- `static INLINE CONST vect_t set (const scalar_t x0, const scalar_t x1, const scalar_t x2, const scalar_t x3, const scalar_t x4, const scalar_t x5, const scalar_t x6, const scalar_t x7)`
- `static INLINE CONST vect_t set (const scalar_t x0, const scalar_t x1, const scalar_t x2, const scalar_t x3)`
- `template<class T >`  
`static INLINE PURE vect_t gather (const scalar_t *const p, const T *const idx)`
- `static INLINE PURE vect_t load (const scalar_t *const p)`
- `static INLINE PURE vect_t loadu (const scalar_t *const p)`
- `static INLINE void store (scalar_t *p, vect_t v)`
- `template<uint8_t k>`  
`static INLINE void maskstore (scalar_t *p, vect_t v)`
- `static INLINE void storeu (scalar_t *p, vect_t v)`
- `static INLINE void stream (scalar_t *p, const vect_t v)`
- `template<int s>`  
`static INLINE CONST vect_t sll (const vect_t a)`
- `template<int s>`  
`static INLINE CONST vect_t srl (const vect_t a)`
- `template<uint8_t s>`  
`static INLINE CONST vect_t shuffle (const vect_t a)`
- `static INLINE CONST vect_t unpacklo_intrinsic (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t unpackhi_intrinsic (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t unpacklo (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t unpackhi (const vect_t a, const vect_t b)`
- `static INLINE void unpacklohi (vect_t &lo, vect_t &hi, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t pack_even (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t pack_odd (const vect_t a, const vect_t b)`
- `static INLINE void pack (vect_t &even, vect_t &odd, const vect_t a, const vect_t b)`
- `static INLINE void transpose (vect_t &r0, vect_t &r1, vect_t &r2, vect_t &r3, vect_t &r4, vect_t &r5, vect_t &r6, vect_t &r7)`
- `template<uint8_t s>`  
`static INLINE CONST vect_t blend (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t add (const vect_t a, const vect_t b)`
- `static INLINE vect_t addin (vect_t &a, const vect_t b)`
- `static INLINE CONST vect_t sub (const vect_t a, const vect_t b)`
- `static INLINE vect_t subin (vect_t &a, const vect_t b)`
- `static INLINE CONST vect_t mul (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmaddin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fnmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fnmaddin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmsub (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmsubin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t round (const vect_t a)`
- `static INLINE CONST vect_t mask_high ()`
- `static INLINE CONST vect_t mulhi_fast (vect_t x, vect_t y)`
- `static INLINE vect_t mod (vect_t &C, const __m512d &P, const __m512d &INVP, const __m512d &NEGP, const vect_t &POW50REM, const __m512d &MIN, const __m512d &MAX, __m512d &Q, __m512d &T)`
- `static INLINE CONST vect_t zero ()`
- `static INLINE CONST vect_t vor (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vxor (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vand (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vandnot (const vect_t a, const vect_t b)`



**Static Public Attributes**

- static const constexpr size\_t `vect_size` = 8
- static const constexpr size\_t `alignment` = 64

**Static Protected Member Functions**

- static `INLINE CONST vect_t signbits` (const `vect_t` x)

**12.298.1 Member Typedef Documentation****12.298.1.1 scalar\_t**

```
using scalar_t = uint64_t
```

**12.298.1.2 aligned\_allocator**

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

**12.298.1.3 aligned\_vector**

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

**12.298.1.4 is\_same\_element**

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

**12.298.1.5 simdHalf**

```
using simdHalf = Simd256<scalar_t>
```

**12.298.1.6 vect\_t**

```
using vect_t = __m512i [inherited]
```

**12.298.1.7 half\_t**

```
using half_t = __m256i [inherited]
```

**12.298.2 Member Function Documentation****12.298.2.1 type\_string()**

```
static const std::string type_string () [inline], [static]
```

**12.298.2.2 set1() [1/2]**

```
static INLINE CONST vect_t set1 (
    const scalar_t x) [inline], [static]
```

**12.298.2.3 set() [1/3]**

```
static INLINE CONST vect_t set (
    const scalar_t x0,
    const scalar_t x1,
    const scalar_t x2,
    const scalar_t x3,
    const scalar_t x4,
    const scalar_t x5,
    const scalar_t x6,
    const scalar_t x7) [inline], [static]
```

**12.298.2.4 gather()** [1/2]

```
template<class T >
static INLINE PURE vect_t gather (
    const scalar_t *const p,
    const T *const idx) [inline], [static]
```

**12.298.2.5 load()** [1/2]

```
static INLINE PURE vect_t load (
    const scalar_t *const p) [inline], [static]
```

**12.298.2.6 loadu()** [1/2]

```
static INLINE PURE vect_t loadu (
    const scalar_t *const p) [inline], [static]
```

**12.298.2.7 store()** [1/2]

```
static INLINE void store (
    scalar_t * p,
    vect_t v) [inline], [static]
```

**12.298.2.8 maskstore()** [1/2]

```
template<uint8_t k>
static INLINE void maskstore (
    scalar_t * p,
    vect_t v) [inline], [static]
```

**12.298.2.9 storeu()** [1/2]

```
static INLINE void storeu (
    scalar_t * p,
    vect_t v) [inline], [static]
```

**12.298.2.10 stream()** [1/2]

```
static INLINE void stream (
    scalar_t * p,
    const vect_t v) [inline], [static]
```

**12.298.2.11 sra()**

```
template<int s>
static INLINE CONST vect_t sra (
    const vect_t a) [inline], [static]
```

**12.298.2.12 greater()**

```
static INLINE CONST vect_t greater (
    vect_t a,
    vect_t b) [inline], [static]
```

**12.298.2.13 lesser()**

```
static INLINE CONST vect_t lesser (
    vect_t a,
    vect_t b) [inline], [static]
```

**12.298.2.14 greater\_eq()**

```
static INLINE CONST vect_t greater_eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.298.2.15 lesser\_eq()**

```
static INLINE CONST vect_t lesser_eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.298.2.16 mullo()**

```
static INLINE CONST vect_t mullo (  
    vect_t a,  
    vect_t b) [inline], [static]
```

**12.298.2.17 mulhi()**

```
static INLINE CONST vect_t mulhi (  
    vect_t a,  
    vect_t b) [inline], [static]
```

**12.298.2.18 mulx()**

```
static INLINE CONST vect_t mulx (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.298.2.19 fmaddx()**

```
static INLINE CONST vect_t fmaddx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.298.2.20 fmaddxin()**

```
static INLINE vect_t fmaddxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.298.2.21 fnmaddx()**

```
static INLINE CONST vect_t fnmaddx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.298.2.22 fnmaddxin()**

```
static INLINE vect_t fnmaddxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.298.2.23 fmsubx()**

```
static INLINE CONST vect_t fmsubx (
    const vect_t c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.298.2.24 fmsubxin()**

```
static INLINE vect_t fmsubxin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.298.2.25 hadd\_to\_scal()**

```
static INLINE CONST scalar_t hadd_to_scal (
    const vect_t a) [inline], [static]
```

**12.298.2.26 valid()**

```
template<class T >
static constexpr bool valid (
    T * p) [inline], [static], [constexpr], [inherited]
```

**12.298.2.27 compliant()**

```
template<class T >
static constexpr bool compliant (
    T n) [inline], [static], [constexpr], [inherited]
```

**12.298.2.28 set1() [2/2]**

```
static INLINE CONST vect_t set1 (
    const scalar_t x) [inline], [static], [inherited]
```

**12.298.2.29 set() [2/3]**

```
static INLINE CONST vect_t set (
    const scalar_t x0,
    const scalar_t x1,
    const scalar_t x2,
    const scalar_t x3,
    const scalar_t x4,
    const scalar_t x5,
    const scalar_t x6,
    const scalar_t x7) [inline], [static], [inherited]
```

**12.298.2.30 set() [3/3]**

```
static INLINE CONST vect_t set (
    const scalar_t x0,
    const scalar_t x1,
    const scalar_t x2,
    const scalar_t x3) [inline], [static], [inherited]
```

**12.298.2.31 gather() [2/2]**

```
template<class T >
static INLINE PURE vect_t gather (
```

```
const scalar_t *const p,
const T *const idx) [inline], [static], [inherited]
```

**12.298.2.32 load() [2/2]**

```
static INLINE PURE vect_t load (
const scalar_t *const p) [inline], [static], [inherited]
```

**12.298.2.33 loadu() [2/2]**

```
static INLINE PURE vect_t loadu (
const scalar_t *const p) [inline], [static], [inherited]
```

**12.298.2.34 store() [2/2]**

```
static INLINE void store (
scalar_t * p,
vect_t v) [inline], [static], [inherited]
```

**12.298.2.35 maskstore() [2/2]**

```
template<uint8_t k>
static INLINE void maskstore (
scalar_t * p,
vect_t v) [inline], [static], [inherited]
```

**12.298.2.36 storeu() [2/2]**

```
static INLINE void storeu (
scalar_t * p,
vect_t v) [inline], [static], [inherited]
```

**12.298.2.37 stream() [2/2]**

```
static INLINE void stream (
scalar_t * p,
const vect_t v) [inline], [static], [inherited]
```

**12.298.2.38 sll()**

```
template<int s>
static INLINE CONST vect_t sll (
const vect_t a) [inline], [static], [inherited]
```

**12.298.2.39 srl()**

```
template<int s>
static INLINE CONST vect_t srl (
const vect_t a) [inline], [static], [inherited]
```

**12.298.2.40 shuffle()**

```
template<uint8_t s>
static INLINE CONST vect_t shuffle (
const vect_t a) [inline], [static], [inherited]
```

**12.298.2.41 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.298.2.42 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.298.2.43 unpacklo()**

```
static INLINE CONST vect_t unpacklo (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.298.2.44 unpackhi()**

```
static INLINE CONST vect_t unpackhi (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.298.2.45 unpacklohi()**

```
static INLINE void unpacklohi (  
    vect_t & lo,  
    vect_t & hi,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.298.2.46 pack\_even()**

```
static INLINE CONST vect_t pack_even (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.298.2.47 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.298.2.48 pack()**

```
static INLINE void pack (  
    vect_t & even,  
    vect_t & odd,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

**12.298.2.49 transpose()**

```
static INLINE void transpose (  
    vect_t & r0,  
    vect_t & r1,  
    vect_t & r2,  
    vect_t & r3,  
    vect_t & r4,
```

```
    vect_t & r5,  
    vect_t & r6,  
    vect_t & r7) [inline], [static], [inherited]
```

#### 12.298.2.50 blend()

```
template<uint8_t s>  
static INLINE CONST vect_t blend (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

#### 12.298.2.51 add()

```
static INLINE CONST vect_t add (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

#### 12.298.2.52 addin()

```
static INLINE vect_t addin (  
    vect_t & a,  
    const vect_t b) [inline], [static], [inherited]
```

#### 12.298.2.53 sub()

```
static INLINE CONST vect_t sub (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

#### 12.298.2.54 subin()

```
static INLINE vect_t subin (  
    vect_t & a,  
    const vect_t b) [inline], [static], [inherited]
```

#### 12.298.2.55 mul()

```
static INLINE CONST vect_t mul (  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

#### 12.298.2.56 fmadd()

```
static INLINE CONST vect_t fmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

#### 12.298.2.57 fmaddin()

```
static INLINE vect_t fmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static], [inherited]
```

#### 12.298.2.58 fnmadd()

```
static INLINE CONST vect_t fnmadd (  
    const vect_t c,
```

```
const vect_t a,
const vect_t b) [inline], [static], [inherited]
```

#### 12.298.2.59 fnmaddin()

```
static INLINE vect_t fnmaddin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

#### 12.298.2.60 fmsub()

```
static INLINE CONST vect_t fmsub (
    const vect_t c,
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

#### 12.298.2.61 fmsubin()

```
static INLINE vect_t fmsubin (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

#### 12.298.2.62 eq()

```
static INLINE CONST vect_t eq (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

#### 12.298.2.63 round()

```
static INLINE CONST vect_t round (
    const vect_t a) [inline], [static], [inherited]
```

#### 12.298.2.64 mask\_high()

```
static INLINE CONST vect_t mask_high () [inline], [static], [inherited]
```

#### 12.298.2.65 mulhi\_fast()

```
INLINE CONST vect_t mulhi_fast (
    vect_t x,
    vect_t y) [static], [inherited]
```

#### 12.298.2.66 mod()

```
INLINE vect_t mod (
    vect_t & C,
    const __m512d & P,
    const __m512d & INVP,
    const __m512d & NEGP,
    const vect_t & POW50REM,
    const __m512d & MIN,
    const __m512d & MAX,
    __m512d & Q,
    __m512d & T) [static], [inherited]
```



**12.298.2.67 signbits()**

```
static INLINE CONST vect_t signbits (
    const vect_t x) [inline], [static], [protected], [inherited]
```

**12.298.2.68 zero()**

```
static INLINE CONST vect_t zero () [inline], [static], [inherited]
```

**12.298.2.69 vor()**

```
static INLINE CONST vect_t vor (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.298.2.70 vxor()**

```
static INLINE CONST vect_t vxor (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.298.2.71 vand()**

```
static INLINE CONST vect_t vand (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.298.2.72 vandnot()**

```
static INLINE CONST vect_t vandnot (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.298.3 Field Documentation****12.298.3.1 vect\_size**

```
const constexpr size_t vect_size = 8 [static], [constexpr], [inherited]
```

**12.298.3.2 alignment**

```
const constexpr size_t alignment = 64 [static], [constexpr], [inherited]
```

The documentation for this struct was generated from the following file:

- [simd512\\_int64.inl](#)

**12.299 Simd512\_impl< true, true, true, 8 > Struct Reference**

Inheritance diagram for Simd512\_impl< true, true, true, 8 >:



## Data Structures

- union [Converter](#)

## Public Types

- using [vect\\_t](#) = \_\_m512i
- using [half\\_t](#) = \_\_m256i
- using [scalar\\_t](#) = int64\_t
- using [simdHalf](#) = Simd256<[scalar\\_t](#)>
- using [aligned\\_allocator](#) = AlignedAllocator<[scalar\\_t](#), Alignment([alignment](#))>
- using [aligned\\_vector](#) = std::vector<[scalar\\_t](#), [aligned\\_allocator](#)>
- template<class [Field](#) >  
using [is\\_same\\_element](#) = std::is\_same<typename [Field::Element](#), [scalar\\_t](#)>

## Static Public Member Functions

- static const std::string [type\\_string](#) ()
- template<class T >  
static constexpr bool [valid](#) (T \*p)
- template<class T >  
static constexpr bool [compliant](#) (T n)
- static [INLINE CONST vect\\_t set1](#) (const [scalar\\_t](#) x)
- static [INLINE CONST vect\\_t set](#) (const [scalar\\_t](#) x0, const [scalar\\_t](#) x1, const [scalar\\_t](#) x2, const [scalar\\_t](#) x3, const [scalar\\_t](#) x4, const [scalar\\_t](#) x5, const [scalar\\_t](#) x6, const [scalar\\_t](#) x7)
- static [INLINE CONST vect\\_t set](#) (const [scalar\\_t](#) x0, const [scalar\\_t](#) x1, const [scalar\\_t](#) x2, const [scalar\\_t](#) x3)
- template<class T >  
static [INLINE PURE vect\\_t gather](#) (const [scalar\\_t](#) \*const p, const T \*const idx)
- static [INLINE PURE vect\\_t load](#) (const [scalar\\_t](#) \*const p)
- static [INLINE PURE vect\\_t loadu](#) (const [scalar\\_t](#) \*const p)
- static [INLINE void store](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)
- template<uint8\_t k>  
static [INLINE void maskstore](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)
- static [INLINE void storeu](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)
- static [INLINE void stream](#) ([scalar\\_t](#) \*p, const [vect\\_t](#) v)
- template<int s>  
static [INLINE CONST vect\\_t sll](#) (const [vect\\_t](#) a)
- template<int s>  
static [INLINE CONST vect\\_t srl](#) (const [vect\\_t](#) a)
- template<int s>  
static [INLINE CONST vect\\_t sra](#) (const [vect\\_t](#) a)
- template<uint8\_t s>  
static [INLINE CONST vect\\_t shuffle](#) (const [vect\\_t](#) a)
- static [INLINE CONST vect\\_t unpacklo\\_intrinsic](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpackhi\\_intrinsic](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpacklo](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t unpackhi](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE void unpacklohi](#) ([vect\\_t](#) &lo, [vect\\_t](#) &hi, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t pack\\_even](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t pack\\_odd](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE void pack](#) ([vect\\_t](#) &even, [vect\\_t](#) &odd, const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE void transpose](#) ([vect\\_t](#) &r0, [vect\\_t](#) &r1, [vect\\_t](#) &r2, [vect\\_t](#) &r3, [vect\\_t](#) &r4, [vect\\_t](#) &r5, [vect\\_t](#) &r6, [vect\\_t](#) &r7)
- template<uint8\_t s>  
static [INLINE CONST vect\\_t blend](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE CONST vect\\_t add](#) (const [vect\\_t](#) a, const [vect\\_t](#) b)
- static [INLINE vect\\_t addin](#) ([vect\\_t](#) &a, const [vect\\_t](#) b)

- static `INLINE CONST vect_t sub` (const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t subin` (`vect_t` &a, const `vect_t` b)
- static `INLINE CONST vect_t mullo` (`vect_t` a, `vect_t` b)
- static `INLINE CONST vect_t mul` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t mulhi` (`vect_t` a, `vect_t` b)
- static `INLINE CONST vect_t mulx` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmadd` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fmaddin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmaddx` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fmaddxin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fnmadd` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fnmaddin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fnmaddx` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fnmaddxin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmsub` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fmsubin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t fmsubx` (const `vect_t` c, const `vect_t` a, const `vect_t` b)
- static `INLINE vect_t fmsubxin` (`vect_t` &c, const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t greater` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t lesser` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t greater_eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t lesser_eq` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST scalar_t hadd_to_scal` (const `vect_t` a)
- static `INLINE CONST vect_t round` (const `vect_t` a)
- static `INLINE CONST vect_t mask_high` ()
- static `INLINE CONST vect_t mulhi_fast` (`vect_t` x, `vect_t` y)
- static `INLINE vect_t mod` (`vect_t` &C, const `__m512d` &P, const `__m512d` &INVP, const `__m512d` &NEGP, const `vect_t` &POW50REM, const `__m512d` &MIN, const `__m512d` &MAX, `__m512d` &Q, `__m512d` &T)
- static `INLINE CONST vect_t zero` ()
- static `INLINE CONST vect_t vor` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t vxor` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t vand` (const `vect_t` a, const `vect_t` b)
- static `INLINE CONST vect_t vandnot` (const `vect_t` a, const `vect_t` b)

### Static Public Attributes

- static const constexpr size\_t `vect_size` = 8
- static const constexpr size\_t `alignment` = 64

### Static Protected Member Functions

- static `INLINE CONST vect_t signbits` (const `vect_t` x)

## 12.299.1 Member Typedef Documentation

### 12.299.1.1 vect\_t

```
using vect_t = __m512i
```

### 12.299.1.2 half\_t

```
using half_t = __m256i
```

### 12.299.1.3 scalar\_t

```
using scalar_t = int64_t
```

**12.299.1.4 simdHalf**

```
using simdHalf = Simd256<scalar_t>
```

**12.299.1.5 aligned\_allocator**

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

**12.299.1.6 aligned\_vector**

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

**12.299.1.7 is\_same\_element**

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

**12.299.2 Member Function Documentation****12.299.2.1 type\_string()**

```
static const std::string type_string () [inline], [static]
```

**12.299.2.2 valid()**

```
template<class T >
static constexpr bool valid (
    T * p) [inline], [static], [constexpr]
```

**12.299.2.3 compliant()**

```
template<class T >
static constexpr bool compliant (
    T n) [inline], [static], [constexpr]
```

**12.299.2.4 set1()**

```
static INLINE CONST vect_t set1 (
    const scalar_t x) [inline], [static]
```

**12.299.2.5 set() [1/2]**

```
static INLINE CONST vect_t set (
    const scalar_t x0,
    const scalar_t x1,
    const scalar_t x2,
    const scalar_t x3,
    const scalar_t x4,
    const scalar_t x5,
    const scalar_t x6,
    const scalar_t x7) [inline], [static]
```

**12.299.2.6 set() [2/2]**

```
static INLINE CONST vect_t set (
    const scalar_t x0,
    const scalar_t x1,
    const scalar_t x2,
    const scalar_t x3) [inline], [static]
```

### 12.299.2.7 gather()

```
template<class T >
static INLINE PURE vect_t gather (
    const scalar_t *const p,
    const T *const idx) [inline], [static]
```

### 12.299.2.8 load()

```
static INLINE PURE vect_t load (
    const scalar_t *const p) [inline], [static]
```

### 12.299.2.9 loadu()

```
static INLINE PURE vect_t loadu (
    const scalar_t *const p) [inline], [static]
```

### 12.299.2.10 store()

```
static INLINE void store (
    scalar_t * p,
    vect_t v) [inline], [static]
```

### 12.299.2.11 maskstore()

```
template<uint8_t k>
static INLINE void maskstore (
    scalar_t * p,
    vect_t v) [inline], [static]
```

### 12.299.2.12 storeu()

```
static INLINE void storeu (
    scalar_t * p,
    vect_t v) [inline], [static]
```

### 12.299.2.13 stream()

```
static INLINE void stream (
    scalar_t * p,
    const vect_t v) [inline], [static]
```

### 12.299.2.14 sll()

```
template<int s>
static INLINE CONST vect_t sll (
    const vect_t a) [inline], [static]
```

### 12.299.2.15 srl()

```
template<int s>
static INLINE CONST vect_t srl (
    const vect_t a) [inline], [static]
```

### 12.299.2.16 sra()

```
template<int s>
static INLINE CONST vect_t sra (
    const vect_t a) [inline], [static]
```

**12.299.2.17 shuffle()**

```
template<uint8_t s>
static INLINE CONST vect_t shuffle (
    const vect_t a) [inline], [static]
```

**12.299.2.18 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.299.2.19 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.299.2.20 unpacklo()**

```
static INLINE CONST vect_t unpacklo (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.299.2.21 unpackhi()**

```
static INLINE CONST vect_t unpackhi (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.299.2.22 unpacklohi()**

```
static INLINE void unpacklohi (
    vect_t & lo,
    vect_t & hi,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.299.2.23 pack\_even()**

```
static INLINE CONST vect_t pack_even (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.299.2.24 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.299.2.25 pack()**

```
static INLINE void pack (
    vect_t & even,
    vect_t & odd,
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.299.2.26 transpose()**

```
static INLINE void transpose (  
    vect_t & r0,  
    vect_t & r1,  
    vect_t & r2,  
    vect_t & r3,  
    vect_t & r4,  
    vect_t & r5,  
    vect_t & r6,  
    vect_t & r7) [inline], [static]
```

**12.299.2.27 blend()**

```
template<uint8_t s>  
static INLINE CONST vect_t blend (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.299.2.28 add()**

```
static INLINE CONST vect_t add (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.299.2.29 addin()**

```
static INLINE vect_t addin (  
    vect_t & a,  
    const vect_t b) [inline], [static]
```

**12.299.2.30 sub()**

```
static INLINE CONST vect_t sub (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.299.2.31 subin()**

```
static INLINE vect_t subin (  
    vect_t & a,  
    const vect_t b) [inline], [static]
```

**12.299.2.32 mullo()**

```
static INLINE CONST vect_t mullo (  
    vect_t a,  
    vect_t b) [inline], [static]
```

**12.299.2.33 mul()**

```
static INLINE CONST vect_t mul (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.299.2.34 mulhi()**

```
static INLINE CONST vect_t mulhi (  
    vect_t a,  
    vect_t b) [inline], [static]
```

**12.299.2.35 mulx()**

```
static INLINE CONST vect_t mulx (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.299.2.36 fmadd()**

```
static INLINE CONST vect_t fmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.299.2.37 fmaddin()**

```
static INLINE vect_t fmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.299.2.38 fmaddx()**

```
static INLINE CONST vect_t fmaddx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.299.2.39 fmaddxin()**

```
static INLINE vect_t fmaddxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.299.2.40 fnmadd()**

```
static INLINE CONST vect_t fnmadd (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.299.2.41 fnmaddin()**

```
static INLINE vect_t fnmaddin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.299.2.42 fnmaddx()**

```
static INLINE CONST vect_t fnmaddx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.299.2.43 fnmaddxin()**

```
static INLINE vect_t fnmaddxin (  
    vect_t & c,
```



```
const vect_t a,  
const vect_t b) [inline], [static]
```

#### 12.299.2.44 fmsub()

```
static INLINE CONST vect_t fmsub (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.299.2.45 fmsubin()

```
static INLINE vect_t fmsubin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.299.2.46 fmsubx()

```
static INLINE CONST vect_t fmsubx (  
    const vect_t c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.299.2.47 fmsubxin()

```
static INLINE vect_t fmsubxin (  
    vect_t & c,  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.299.2.48 eq()

```
static INLINE CONST vect_t eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.299.2.49 greater()

```
static INLINE CONST vect_t greater (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.299.2.50 lesser()

```
static INLINE CONST vect_t lesser (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.299.2.51 greater\_eq()

```
static INLINE CONST vect_t greater_eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

#### 12.299.2.52 lesser\_eq()

```
static INLINE CONST vect_t lesser_eq (  
    const vect_t a,  
    const vect_t b) [inline], [static]
```

**12.299.2.53 hadd\_to\_scal()**

```
static INLINE CONST scalar_t hadd_to_scal (
    const vect_t a)  [inline], [static]
```

**12.299.2.54 round()**

```
static INLINE CONST vect_t round (
    const vect_t a)  [inline], [static]
```

**12.299.2.55 mask\_high()**

```
static INLINE CONST vect_t mask_high ()  [inline], [static]
```

**12.299.2.56 mulhi\_fast()**

```
INLINE CONST vect_t mulhi_fast (
    vect_t x,
    vect_t y)  [static]
```

**12.299.2.57 mod()**

```
INLINE vect_t mod (
    vect_t & C,
    const __m512d & P,
    const __m512d & INVP,
    const __m512d & NEGP,
    const vect_t & POW50REM,
    const __m512d & MIN,
    const __m512d & MAX,
    __m512d & Q,
    __m512d & T)  [static]
```

**12.299.2.58 signbits()**

```
static INLINE CONST vect_t signbits (
    const vect_t x)  [inline], [static], [protected]
```

**12.299.2.59 zero()**

```
static INLINE CONST vect_t zero ()  [inline], [static], [inherited]
```

**12.299.2.60 vor()**

```
static INLINE CONST vect_t vor (
    const vect_t a,
    const vect_t b)  [inline], [static], [inherited]
```

**12.299.2.61 vxor()**

```
static INLINE CONST vect_t vxor (
    const vect_t a,
    const vect_t b)  [inline], [static], [inherited]
```

**12.299.2.62 vand()**

```
static INLINE CONST vect_t vand (
    const vect_t a,
    const vect_t b)  [inline], [static], [inherited]
```

**12.299.2.63 vandnot()**

```
static INLINE CONST vect_t vandnot (
    const vect_t a,
    const vect_t b) [inline], [static], [inherited]
```

**12.299.3 Field Documentation****12.299.3.1 vect\_size**

```
const constexpr size_t vect_size = 8 [static], [constexpr]
```

**12.299.3.2 alignment**

```
const constexpr size_t alignment = 64 [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [simd512\\_int64.inl](#)

**12.300 Simd512i\_base Struct Reference**

Inheritance diagram for Simd512i\_base:

**Public Types**

- using `vect_t` = `__m512i`

**Static Public Member Functions**

- static `INLINE` `CONST` `vect_t` `zero` ()
- static `INLINE` `CONST` `vect_t` `vor` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `vxor` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `vand` (const `vect_t` a, const `vect_t` b)
- static `INLINE` `CONST` `vect_t` `vandnot` (const `vect_t` a, const `vect_t` b)

**12.300.1 Member Typedef Documentation****12.300.1.1 vect\_t**

```
using vect_t = __m512i
```

**12.300.2 Member Function Documentation****12.300.2.1 zero()**

```
static INLINE CONST vect_t zero () [inline], [static]
```

**12.300.2.2 vor()**

```
static INLINE CONST vect_t vor (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.300.2.3 vxor()**

```
static INLINE CONST vect_t vxor (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.300.2.4 vand()**

```
static INLINE CONST vect_t vand (
    const vect_t a,
    const vect_t b) [inline], [static]
```

**12.300.2.5 vandnot()**

```
static INLINE CONST vect_t vandnot (
    const vect_t a,
    const vect_t b) [inline], [static]
```

The documentation for this struct was generated from the following file:

- [simd512.inl](#)

**12.301 SimdChooser< T, bool, bool > Struct Template Reference**

```
#include <fflas_simd.h>
```

The documentation for this struct was generated from the following file:

- [fflas\\_simd.h](#)

**12.302 SimdChooser< T, false, b > Struct Template Reference**

```
#include <fflas_simd.h>
```

**Public Types**

- using [value](#) = [NoSimd](#)<T>

**12.302.1 Member Typedef Documentation****12.302.1.1 value**

```
template<class T , bool b>
using value = NoSimd<T>
```

The documentation for this struct was generated from the following file:

- [fflas\\_simd.h](#)

**12.303 SimdChooser< T, true, false > Struct Template Reference**

```
#include <fflas_simd.h>
```

**Public Types**

- using [value](#) = [NoSimd](#)<T>

### 12.303.1 Member Typedef Documentation

#### 12.303.1.1 value

```
template<class T >
using value = NoSimd<T>
```

The documentation for this struct was generated from the following file:

- [fflas\\_simd.h](#)

## 12.304 SimdChooser< T, true, true > Struct Template Reference

```
#include <fflas_simd.h>
```

### Public Types

- using [value](#) = [NoSimd](#)<T>

### 12.304.1 Member Typedef Documentation

#### 12.304.1.1 value

```
template<class T >
using value = NoSimd<T>
```

The documentation for this struct was generated from the following file:

- [fflas\\_simd.h](#)

## 12.305 simdToType< T > Struct Template Reference

The documentation for this struct was generated from the following file:

- [fflas\\_simd.h](#)

## 12.306 Single Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.307 Sparse< Field, SparseMatrix\_t, IdxT, PtrT > Struct Template Reference

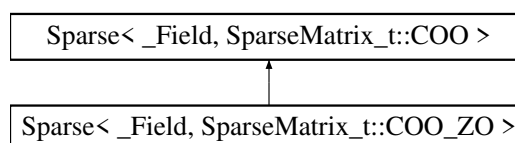
The documentation for this struct was generated from the following file:

- [fflas\\_sparse.h](#)

## 12.308 Sparse< \_Field, SparseMatrix\_t::COO > Struct Template Reference

```
#include <coo.h>
```

Inheritance diagram for Sparse< \_Field, SparseMatrix\_t::COO >:



## Public Types

- using `Field` = `_Field`

## Data Fields

- `index_t` \* `col` = `nullptr`
- `index_t` \* `row` = `nullptr`
- `_Field::Element_ptr` `dat`
- bool `delayed` = `false`
- `uint64_t` `kmax` = `0`
- `index_t` `m` = `0`
- `index_t` `n` = `0`
- `uint64_t` `nnz` = `0`
- `uint64_t` `nElements` = `0`
- `uint64_t` `maxrow` = `0`

## 12.308.1 Member Typedef Documentation

### 12.308.1.1 Field

```
template<class _Field >
using Field = _Field
```

## 12.308.2 Field Documentation

### 12.308.2.1 col

```
template<class _Field >
index_t* col = nullptr
```

### 12.308.2.2 row

```
template<class _Field >
index_t* row = nullptr
```

### 12.308.2.3 dat

```
template<class _Field >
_Field::Element_ptr dat
```

### 12.308.2.4 delayed

```
template<class _Field >
bool delayed = false
```

### 12.308.2.5 kmax

```
template<class _Field >
uint64_t kmax = 0
```

### 12.308.2.6 m

```
template<class _Field >
index_t m = 0
```

### 12.308.2.7 n

```
template<class _Field >
index_t n = 0
```

**12.308.2.8 nnz**

```
template<class _Field >
uint64_t nnz = 0
```

**12.308.2.9 nElements**

```
template<class _Field >
uint64_t nElements = 0
```

**12.308.2.10 maxrow**

```
template<class _Field >
uint64_t maxrow = 0
```

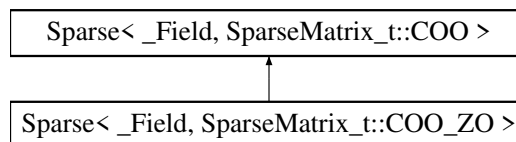
The documentation for this struct was generated from the following file:

- [coo.h](#)

## 12.309 Sparse< \_Field, SparseMatrix\_t::COO\_ZO > Struct Template Reference

```
#include <coo.h>
```

Inheritance diagram for Sparse< \_Field, SparseMatrix\_t::COO\_ZO >:

**Public Types**

- using [Field](#) = \_Field

**Data Fields**

- \_Field::Element [cst](#) = 1
- [index\\_t](#) \* [col](#) = nullptr
- [index\\_t](#) \* [row](#) = nullptr
- \_Field::Element\_ptr [dat](#)
- bool [delayed](#) = false
- uint64\_t [kmax](#) = 0
- [index\\_t](#) [m](#) = 0
- [index\\_t](#) [n](#) = 0
- uint64\_t [nnz](#) = 0
- uint64\_t [nElements](#) = 0
- uint64\_t [maxrow](#) = 0

**12.309.1 Member Typedef Documentation****12.309.1.1 Field**

```
template<class _Field >
using Field = _Field
```

## 12.309.2 Field Documentation

### 12.309.2.1 cst

```
template<class _Field >
_Field::Element cst = 1
```

### 12.309.2.2 col

```
template<class _Field >
index_t* col = nullptr [inherited]
```

### 12.309.2.3 row

```
template<class _Field >
index_t* row = nullptr [inherited]
```

### 12.309.2.4 dat

```
template<class _Field >
_Field::Element_ptr dat [inherited]
```

### 12.309.2.5 delayed

```
template<class _Field >
bool delayed = false [inherited]
```

### 12.309.2.6 kmax

```
template<class _Field >
uint64_t kmax = 0 [inherited]
```

### 12.309.2.7 m

```
template<class _Field >
index_t m = 0 [inherited]
```

### 12.309.2.8 n

```
template<class _Field >
index_t n = 0 [inherited]
```

### 12.309.2.9 nnz

```
template<class _Field >
uint64_t nnz = 0 [inherited]
```

### 12.309.2.10 nElements

```
template<class _Field >
uint64_t nElements = 0 [inherited]
```

### 12.309.2.11 maxrow

```
template<class _Field >
uint64_t maxrow = 0 [inherited]
```

The documentation for this struct was generated from the following file:

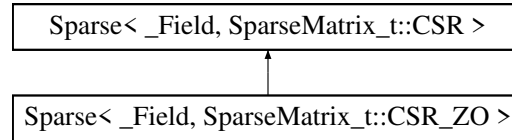
- [coo.h](#)



## 12.310 Sparse< \_Field, SparseMatrix\_t::CSR > Struct Template Reference

```
#include <csr.h>
```

Inheritance diagram for Sparse< \_Field, SparseMatrix\_t::CSR >:



### Public Types

- using [Field](#) = \_Field

### Data Fields

- bool [delayed](#) = false
- uint64\_t [kmax](#) = 0
- [index\\_t](#) [m](#) = 0
- [index\\_t](#) [n](#) = 0
- uint64\_t [nnz](#) = 0
- uint64\_t [nElements](#) = 0
- uint64\_t [maxrow](#) = 0
- [index\\_t](#) \* [col](#) = nullptr
- [index\\_t](#) \* [st](#) = nullptr
- [index\\_t](#) \* [stend](#) = nullptr
- \_Field::Element\_ptr [dat](#)

### 12.310.1 Member Typedef Documentation

#### 12.310.1.1 Field

```
template<class _Field >
using Field = _Field
```

### 12.310.2 Field Documentation

#### 12.310.2.1 delayed

```
template<class _Field >
bool delayed = false
```

#### 12.310.2.2 kmax

```
template<class _Field >
uint64_t kmax = 0
```

#### 12.310.2.3 m

```
template<class _Field >
index\_t m = 0
```

#### 12.310.2.4 n

```
template<class _Field >
index\_t n = 0
```

**12.310.2.5 nnz**

```
template<class _Field >
uint64_t nnz = 0
```

**12.310.2.6 nElements**

```
template<class _Field >
uint64_t nElements = 0
```

**12.310.2.7 maxrow**

```
template<class _Field >
uint64_t maxrow = 0
```

**12.310.2.8 col**

```
template<class _Field >
index_t* col = nullptr
```

**12.310.2.9 st**

```
template<class _Field >
index_t* st = nullptr
```

**12.310.2.10 stend**

```
template<class _Field >
index_t* stend = nullptr
```

**12.310.2.11 dat**

```
template<class _Field >
_Field::Element_ptr dat
```

The documentation for this struct was generated from the following file:

- [csr.h](#)

## 12.311 Sparse<\_Field, SparseMatrix\_t::CSR\_HYB > Struct Template Reference

```
#include <csr_hyb.h>
```

**Public Types**

- using [Field](#) = \_Field

**Data Fields**

- bool [delayed](#) = false
- [index\\_t](#) \* [col](#) = nullptr
- [index\\_t](#) \* [st](#) = nullptr
- \_Field::Element\_ptr [dat](#)
- uint64\_t [kmax](#) = 0
- [index\\_t](#) [m](#) = 0
- [index\\_t](#) [n](#) = 0
- uint64\_t [nnz](#) = 0
- uint64\_t [nElements](#) = 0

- uint64\_t `maxrow` = 0
- uint64\_t `nOnes` = 0
- uint64\_t `nMOnes` = 0
- uint64\_t `nOthers` = 0

## 12.311.1 Member Typedef Documentation

### 12.311.1.1 Field

```
template<class _Field >  
using Field = _Field
```

## 12.311.2 Field Documentation

### 12.311.2.1 delayed

```
template<class _Field >  
bool delayed = false
```

### 12.311.2.2 col

```
template<class _Field >  
index_t* col = nullptr
```

### 12.311.2.3 st

```
template<class _Field >  
index_t* st = nullptr
```

### 12.311.2.4 dat

```
template<class _Field >  
_Field::Element_ptr dat
```

### 12.311.2.5 kmax

```
template<class _Field >  
uint64_t kmax = 0
```

### 12.311.2.6 m

```
template<class _Field >  
index_t m = 0
```

### 12.311.2.7 n

```
template<class _Field >  
index_t n = 0
```

### 12.311.2.8 nnz

```
template<class _Field >  
uint64_t nnz = 0
```

### 12.311.2.9 nElements

```
template<class _Field >  
uint64_t nElements = 0
```

**12.311.2.10 maxrow**

```
template<class _Field >
uint64_t maxrow = 0
```

**12.311.2.11 nOnes**

```
template<class _Field >
uint64_t nOnes = 0
```

**12.311.2.12 nMOnes**

```
template<class _Field >
uint64_t nMOnes = 0
```

**12.311.2.13 nOthers**

```
template<class _Field >
uint64_t nOthers = 0
```

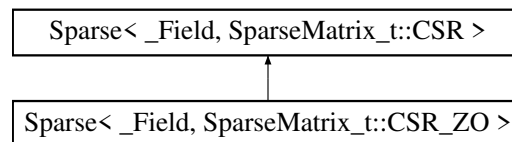
The documentation for this struct was generated from the following file:

- [csr\\_hyb.h](#)

## 12.312 Sparse< \_Field, SparseMatrix\_t::CSR\_ZO > Struct Template Reference

```
#include <csr.h>
```

Inheritance diagram for Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >:

**Public Types**

- using [Field](#) = \_Field

**Data Fields**

- int64\_t [cst](#) = 1
- bool [delayed](#) = false
- uint64\_t [kmax](#) = 0
- [index\\_t](#) [m](#) = 0
- [index\\_t](#) [n](#) = 0
- uint64\_t [nnz](#) = 0
- uint64\_t [nElements](#) = 0
- uint64\_t [maxrow](#) = 0
- [index\\_t](#) \* [col](#) = nullptr
- [index\\_t](#) \* [st](#) = nullptr
- [index\\_t](#) \* [stend](#) = nullptr
- \_Field::Element\_ptr [dat](#)

## 12.312.1 Member Typedef Documentation

### 12.312.1.1 Field

```
template<class _Field >
using Field = _Field
```

## 12.312.2 Field Documentation

### 12.312.2.1 cst

```
template<class _Field >
int64_t cst = 1
```

### 12.312.2.2 delayed

```
template<class _Field >
bool delayed = false
```

### 12.312.2.3 kmax

```
template<class _Field >
uint64_t kmax = 0 [inherited]
```

### 12.312.2.4 m

```
template<class _Field >
index_t m = 0 [inherited]
```

### 12.312.2.5 n

```
template<class _Field >
index_t n = 0 [inherited]
```

### 12.312.2.6 nnz

```
template<class _Field >
uint64_t nnz = 0 [inherited]
```

### 12.312.2.7 nElements

```
template<class _Field >
uint64_t nElements = 0 [inherited]
```

### 12.312.2.8 maxrow

```
template<class _Field >
uint64_t maxrow = 0 [inherited]
```

### 12.312.2.9 col

```
template<class _Field >
index_t* col = nullptr [inherited]
```

### 12.312.2.10 st

```
template<class _Field >
index_t* st = nullptr [inherited]
```

**12.312.2.11 stend**

```
template<class _Field >
index_t* stend = nullptr [inherited]
```

**12.312.2.12 dat**

```
template<class _Field >
_Field::Element_ptr dat [inherited]
```

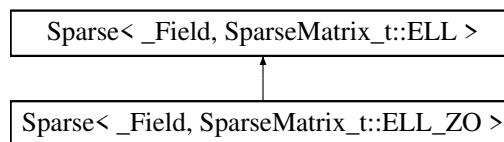
The documentation for this struct was generated from the following file:

- [csr.h](#)

## 12.313 Sparse< \_Field, SparseMatrix\_t::ELL > Struct Template Reference

```
#include <ell.h>
```

Inheritance diagram for Sparse< \_Field, SparseMatrix\_t::ELL >:

**Public Types**

- using [Field](#) = \_Field

**Data Fields**

- bool [delayed](#) = false
- uint64\_t [kmax](#) = 0
- [index\\_t](#) [m](#) = 0
- [index\\_t](#) [n](#) = 0
- [index\\_t](#) [ld](#) = 0
- uint64\_t [nnz](#) = 0
- uint64\_t [nElements](#) = 0
- uint64\_t [maxrow](#) = 0
- [index\\_t](#) \* [col](#) = nullptr
- \_Field::Element\_ptr [dat](#)

**12.313.1 Member Typedef Documentation****12.313.1.1 Field**

```
template<class _Field >
using Field = _Field
```

**12.313.2 Field Documentation****12.313.2.1 delayed**

```
template<class _Field >
bool delayed = false
```

**12.313.2.2 kmax**

```
template<class _Field >
uint64_t kmax = 0
```

**12.313.2.3 m**

```
template<class _Field >
index_t m = 0
```

**12.313.2.4 n**

```
template<class _Field >
index_t n = 0
```

**12.313.2.5 ld**

```
template<class _Field >
index_t ld = 0
```

**12.313.2.6 nnz**

```
template<class _Field >
uint64_t nnz = 0
```

**12.313.2.7 nElements**

```
template<class _Field >
uint64_t nElements = 0
```

**12.313.2.8 maxrow**

```
template<class _Field >
uint64_t maxrow = 0
```

**12.313.2.9 col**

```
template<class _Field >
index_t* col = nullptr
```

**12.313.2.10 dat**

```
template<class _Field >
_Field::Element_ptr dat
```

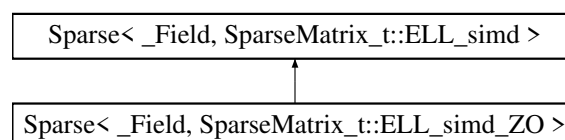
The documentation for this struct was generated from the following file:

- [ell.h](#)

## 12.314 Sparse< \_Field, SparseMatrix\_t::ELL\_simd > Struct Template Reference

```
#include <ell_simd.h>
```

Inheritance diagram for Sparse< \_Field, SparseMatrix\_t::ELL\_simd >:



## Data Fields

- bool `delayed` = false
- int `chunk` = 0
- `index_t` `m` = 0
- `index_t` `n` = 0
- `index_t` `ld` = 0
- `uint64_t` `kmax` = 0
- `uint64_t` `nnz` = 0
- `uint64_t` `nElements` = 0
- `uint64_t` `maxrow` = 0
- `uint64_t` `nChunks` = 0
- `index_t` \* `col` = nullptr
- `_Field::Element_ptr` `dat`

## 12.314.1 Field Documentation

### 12.314.1.1 `delayed`

```
template<class _Field >
bool delayed = false
```

### 12.314.1.2 `chunk`

```
template<class _Field >
int chunk = 0
```

### 12.314.1.3 `m`

```
template<class _Field >
index_t m = 0
```

### 12.314.1.4 `n`

```
template<class _Field >
index_t n = 0
```

### 12.314.1.5 `ld`

```
template<class _Field >
index_t ld = 0
```

### 12.314.1.6 `kmax`

```
template<class _Field >
uint64_t kmax = 0
```

### 12.314.1.7 `nnz`

```
template<class _Field >
uint64_t nnz = 0
```

### 12.314.1.8 `nElements`

```
template<class _Field >
uint64_t nElements = 0
```



**12.314.1.9 maxrow**

```
template<class _Field >
uint64_t maxrow = 0
```

**12.314.1.10 nChunks**

```
template<class _Field >
uint64_t nChunks = 0
```

**12.314.1.11 col**

```
template<class _Field >
index_t* col = nullptr
```

**12.314.1.12 dat**

```
template<class _Field >
_Field::Element_ptr dat
```

The documentation for this struct was generated from the following file:

- [ell\\_simd.h](#)

## 12.315 Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO > Struct Template Reference

```
#include <ell_simd.h>
```

Inheritance diagram for Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >:

**Data Fields**

- \_Field::Element [cst](#) = 1
- bool [delayed](#) = false
- int [chunk](#) = 0
- [index\\_t](#) [m](#) = 0
- [index\\_t](#) [n](#) = 0
- [index\\_t](#) [ld](#) = 0
- uint64\_t [kmax](#) = 0
- uint64\_t [nnz](#) = 0
- uint64\_t [nElements](#) = 0
- uint64\_t [maxrow](#) = 0
- uint64\_t [nChunks](#) = 0
- [index\\_t](#) \* [col](#) = nullptr
- \_Field::Element\_ptr [dat](#)

**12.315.1 Field Documentation****12.315.1.1 cst**

```
template<class _Field >
_Field::Element cst = 1
```

**12.315.1.2 delayed**

```
template<class _Field >
bool delayed = false [inherited]
```

**12.315.1.3 chunk**

```
template<class _Field >
int chunk = 0 [inherited]
```

**12.315.1.4 m**

```
template<class _Field >
index_t m = 0 [inherited]
```

**12.315.1.5 n**

```
template<class _Field >
index_t n = 0 [inherited]
```

**12.315.1.6 ld**

```
template<class _Field >
index_t ld = 0 [inherited]
```

**12.315.1.7 kmax**

```
template<class _Field >
uint64_t kmax = 0 [inherited]
```

**12.315.1.8 nnz**

```
template<class _Field >
uint64_t nnz = 0 [inherited]
```

**12.315.1.9 nElements**

```
template<class _Field >
uint64_t nElements = 0 [inherited]
```

**12.315.1.10 maxrow**

```
template<class _Field >
uint64_t maxrow = 0 [inherited]
```

**12.315.1.11 nChunks**

```
template<class _Field >
uint64_t nChunks = 0 [inherited]
```

**12.315.1.12 col**

```
template<class _Field >
index_t* col = nullptr [inherited]
```

**12.315.1.13 dat**

```
template<class _Field >
_Field::Element_ptr dat [inherited]
```

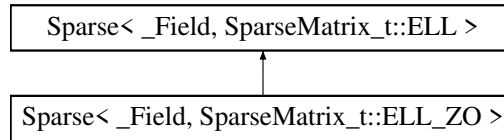
The documentation for this struct was generated from the following file:

- [ell\\_simd.h](#)

## 12.316 Sparse< \_Field, SparseMatrix\_t::ELL\_ZO > Struct Template Reference

```
#include <ell.h>
```

Inheritance diagram for Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >:



### Public Types

- using [Field](#) = \_Field

### Data Fields

- \_Field::Element [cst](#) = 1
- bool [delayed](#) = false
- uint64\_t [kmax](#) = 0
- [index\\_t](#) [m](#) = 0
- [index\\_t](#) [n](#) = 0
- [index\\_t](#) [ld](#) = 0
- uint64\_t [nnz](#) = 0
- uint64\_t [nElements](#) = 0
- uint64\_t [maxrow](#) = 0
- [index\\_t](#) \* [col](#) = nullptr
- \_Field::Element\_ptr [dat](#)

### 12.316.1 Member Typedef Documentation

#### 12.316.1.1 Field

```
template<class _Field >
using Field = _Field
```

### 12.316.2 Field Documentation

#### 12.316.2.1 cst

```
template<class _Field >
_Field::Element cst = 1
```

#### 12.316.2.2 delayed

```
template<class _Field >
bool delayed = false [inherited]
```

#### 12.316.2.3 kmax

```
template<class _Field >
uint64_t kmax = 0 [inherited]
```

#### 12.316.2.4 m

```
template<class _Field >
index\_t m = 0 [inherited]
```

**12.316.2.5 n**

```
template<class _Field >
index_t n = 0 [inherited]
```

**12.316.2.6 ld**

```
template<class _Field >
index_t ld = 0 [inherited]
```

**12.316.2.7 nnz**

```
template<class _Field >
uint64_t nnz = 0 [inherited]
```

**12.316.2.8 nElements**

```
template<class _Field >
uint64_t nElements = 0 [inherited]
```

**12.316.2.9 maxrow**

```
template<class _Field >
uint64_t maxrow = 0 [inherited]
```

**12.316.2.10 col**

```
template<class _Field >
index_t* col = nullptr [inherited]
```

**12.316.2.11 dat**

```
template<class _Field >
_Field::Element_ptr dat [inherited]
```

The documentation for this struct was generated from the following file:

- [ell.h](#)

## 12.317 Sparse<\_Field, SparseMatrix\_t::HYB\_ZO > Struct Template Reference

```
#include <hyb_zo.h>
```

**Public Types**

- using [Field](#) = [\\_Field](#)
- typedef [Sparse](#)< [\\_Field](#), [SparseMatrix\\_t::HYB\\_ZO](#) > [Self\\_t](#)

**Data Fields**

- bool [delayed](#) = false
- uint64\_t [kmax](#) = 0
- [index\\_t](#) [m](#) = 0
- [index\\_t](#) [n](#) = 0
- uint64\_t [nnz](#) = 0
- uint64\_t [maxrow](#) = 0
- uint64\_t [nElements](#) = 0
- [Sparse](#)< [\\_Field](#), [SparseMatrix\\_t::CSR](#) > \* [dat](#) = nullptr
- [Sparse](#)< [\\_Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > \* [one](#) = nullptr
- [Sparse](#)< [\\_Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > \* [mone](#) = nullptr

## 12.317.1 Member Typedef Documentation

### 12.317.1.1 Field

```
template<class _Field >
using Field = _Field
```

### 12.317.1.2 Self\_t

```
template<class _Field >
Sparse<_Field, SparseMatrix_t::HYB_ZO> Self_t
```

## 12.317.2 Field Documentation

### 12.317.2.1 delayed

```
template<class _Field >
bool delayed = false
```

### 12.317.2.2 kmax

```
template<class _Field >
uint64_t kmax = 0
```

### 12.317.2.3 m

```
template<class _Field >
index_t m = 0
```

### 12.317.2.4 n

```
template<class _Field >
index_t n = 0
```

### 12.317.2.5 nnz

```
template<class _Field >
uint64_t nnz = 0
```

### 12.317.2.6 maxrow

```
template<class _Field >
uint64_t maxrow = 0
```

### 12.317.2.7 nElements

```
template<class _Field >
uint64_t nElements = 0
```

### 12.317.2.8 dat

```
template<class _Field >
Sparse<_Field, SparseMatrix_t::CSR>* dat = nullptr
```

### 12.317.2.9 one

```
template<class _Field >
Sparse<_Field, SparseMatrix_t::CSR_ZO>* one = nullptr
```

### 12.317.2.10 mone

```
template<class _Field >
Sparse<_Field, SparseMatrix_t::CSR_ZO>* mone = nullptr
```

The documentation for this struct was generated from the following file:

- [hyb\\_zo.h](#)

## 12.318 Sparse< \_Field, SparseMatrix\_t::SELL > Struct Template Reference

```
#include <sell.h>
Inheritance diagram for Sparse< _Field, SparseMatrix_t::SELL >:
```



### Public Types

- using [Field](#) = [\\_Field](#)

### Data Fields

- bool [delayed](#) = false
- int [chunk](#) = 0
- [index\\_t](#) [kmax](#) = 0
- [index\\_t](#) [m](#) = 0
- [index\\_t](#) [n](#) = 0
- [index\\_t](#) [maxrow](#) = 0
- [index\\_t](#) [sigma](#) = 0
- [index\\_t](#) [nChunks](#) = 0
- [uint64\\_t](#) [nnz](#) = 0
- [uint64\\_t](#) [nElements](#) = 0
- [index\\_t](#) \* [perm](#) = nullptr
- [uint64\\_t](#) \* [st](#) = nullptr
- [index\\_t](#) \* [chunkSize](#) = nullptr
- [index\\_t](#) \* [col](#) = nullptr
- [\\_Field::Element\\_ptr](#) [dat](#)

## 12.318.1 Member Typedef Documentation

### 12.318.1.1 Field

```
template<class _Field >
using Field = \_Field
```

## 12.318.2 Field Documentation

### 12.318.2.1 delayed

```
template<class _Field >
bool delayed = false
```

#### 12.318.2.2 chunk

```
template<class _Field >  
int chunk = 0
```

#### 12.318.2.3 kmax

```
template<class _Field >  
index_t kmax = 0
```

#### 12.318.2.4 m

```
template<class _Field >  
index_t m = 0
```

#### 12.318.2.5 n

```
template<class _Field >  
index_t n = 0
```

#### 12.318.2.6 maxrow

```
template<class _Field >  
index_t maxrow = 0
```

#### 12.318.2.7 sigma

```
template<class _Field >  
index_t sigma = 0
```

#### 12.318.2.8 nChunks

```
template<class _Field >  
index_t nChunks = 0
```

#### 12.318.2.9 nnz

```
template<class _Field >  
uint64_t nnz = 0
```

#### 12.318.2.10 nElements

```
template<class _Field >  
uint64_t nElements = 0
```

#### 12.318.2.11 perm

```
template<class _Field >  
index_t* perm = nullptr
```

#### 12.318.2.12 st

```
template<class _Field >  
uint64_t* st = nullptr
```

#### 12.318.2.13 chunkSize

```
template<class _Field >  
index_t* chunkSize = nullptr
```

**12.318.2.14 col**

```
template<class _Field >
index_t* col = nullptr
```

**12.318.2.15 dat**

```
template<class _Field >
_Field::Element_ptr dat
```

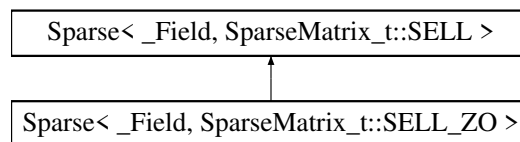
The documentation for this struct was generated from the following file:

- [sell.h](#)

## 12.319 Sparse< \_Field, SparseMatrix\_t::SELL\_ZO > Struct Template Reference

```
#include <sell.h>
```

Inheritance diagram for Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >:

**Public Types**

- using [Field](#) = \_Field

**Data Fields**

- \_Field::Element [cst](#) = 1
- bool [delayed](#) = false
- int [chunk](#) = 0
- index\_t [kmax](#) = 0
- index\_t [m](#) = 0
- index\_t [n](#) = 0
- index\_t [maxrow](#) = 0
- index\_t [sigma](#) = 0
- index\_t [nChunks](#) = 0
- uint64\_t [nnz](#) = 0
- uint64\_t [nElements](#) = 0
- index\_t \* [perm](#) = nullptr
- uint64\_t \* [st](#) = nullptr
- index\_t \* [chunkSize](#) = nullptr
- index\_t \* [col](#) = nullptr
- \_Field::Element\_ptr [dat](#)

**12.319.1 Member Typedef Documentation****12.319.1.1 Field**

```
template<class _Field >
using Field = _Field
```



## 12.319.2 Field Documentation

### 12.319.2.1 cst

```
template<class _Field >
_Field::Element cst = 1
```

### 12.319.2.2 delayed

```
template<class _Field >
bool delayed = false [inherited]
```

### 12.319.2.3 chunk

```
template<class _Field >
int chunk = 0 [inherited]
```

### 12.319.2.4 kmax

```
template<class _Field >
index_t kmax = 0 [inherited]
```

### 12.319.2.5 m

```
template<class _Field >
index_t m = 0 [inherited]
```

### 12.319.2.6 n

```
template<class _Field >
index_t n = 0 [inherited]
```

### 12.319.2.7 maxrow

```
template<class _Field >
index_t maxrow = 0 [inherited]
```

### 12.319.2.8 sigma

```
template<class _Field >
index_t sigma = 0 [inherited]
```

### 12.319.2.9 nChunks

```
template<class _Field >
index_t nChunks = 0 [inherited]
```

### 12.319.2.10 nnz

```
template<class _Field >
uint64_t nnz = 0 [inherited]
```

### 12.319.2.11 nElements

```
template<class _Field >
uint64_t nElements = 0 [inherited]
```

### 12.319.2.12 perm

```
template<class _Field >
index_t* perm = nullptr [inherited]
```

**12.319.2.13 st**

```
template<class _Field >
uint64_t* st = nullptr [inherited]
```

**12.319.2.14 chunkSize**

```
template<class _Field >
index_t* chunkSize = nullptr [inherited]
```

**12.319.2.15 col**

```
template<class _Field >
index_t* col = nullptr [inherited]
```

**12.319.2.16 dat**

```
template<class _Field >
_Field::Element_ptr dat [inherited]
```

The documentation for this struct was generated from the following file:

- [sell.h](#)

**12.320 SpMat< Field, flag > Struct Template Reference**

```
#include <fflas_sparse.h>
```

**Data Fields**

- [FFLAS::CooMat< Field > \\* \\_coo](#) = nullptr
- [FFLAS::CsrMat< Field > \\* \\_csr](#) = nullptr
- [FFLAS::EllMat< Field > \\* \\_ell](#) = nullptr

**12.320.1 Field Documentation****12.320.1.1 \_coo**

```
template<class Field , int flag = HelperFlag::none>
FFLAS::CooMat<Field>* _coo = nullptr
```

**12.320.1.2 \_csr**

```
template<class Field , int flag = HelperFlag::none>
FFLAS::CsrMat<Field>* _csr = nullptr
```

**12.320.1.3 \_ell**

```
template<class Field , int flag = HelperFlag::none>
FFLAS::EllMat<Field>* _ell = nullptr
```

The documentation for this struct was generated from the following file:

- [fflas\\_sparse.h](#)

**12.321 StatsMatrix Struct Reference**

```
#include <utils.h>
```

**Data Fields**

- uint64\_t rowdim = 0
- uint64\_t coldim = 0
- uint64\_t nOnes = 0
- uint64\_t nMOnes = 0
- uint64\_t nOthers = 0
- uint64\_t nnz = 0
- uint64\_t maxRow = 0
- uint64\_t minRow = 0
- uint64\_t averageRow = 0
- uint64\_t deviationRow = 0
- uint64\_t maxCol = 0
- uint64\_t minCol = 0
- uint64\_t averageCol = 0
- uint64\_t deviationCol = 0
- uint64\_t minColDifference = 0
- uint64\_t maxColDifference = 0
- uint64\_t averageColDifference = 0
- uint64\_t deviationColDifference = 0
- uint64\_t minRowDifference = 0
- uint64\_t maxRowDifference = 0
- uint64\_t averageRowDifference = 0
- uint64\_t deviationRowDifference = 0
- uint64\_t nDenseRows = 0
- uint64\_t nDenseCols = 0
- uint64\_t nEmptyRows = 0
- uint64\_t nEmptyCols = 0
- uint64\_t nEmptyColsEnd = 0
- std::vector< uint64\_t > denseRows
- std::vector< uint64\_t > denseCols

**12.321.1 Field Documentation****12.321.1.1 rowdim**

```
uint64_t rowdim = 0
```

**12.321.1.2 coldim**

```
uint64_t coldim = 0
```

**12.321.1.3 nOnes**

```
uint64_t nOnes = 0
```

**12.321.1.4 nMOnes**

```
uint64_t nMOnes = 0
```

**12.321.1.5 nOthers**

```
uint64_t nOthers = 0
```

**12.321.1.6 nnz**

```
uint64_t nnz = 0
```

**12.321.1.7 maxRow**

```
uint64_t maxRow = 0
```

**12.321.1.8 minRow**

```
uint64_t minRow = 0
```

**12.321.1.9 averageRow**

```
uint64_t averageRow = 0
```

**12.321.1.10 deviationRow**

```
uint64_t deviationRow = 0
```

**12.321.1.11 maxCol**

```
uint64_t maxCol = 0
```

**12.321.1.12 minCol**

```
uint64_t minCol = 0
```

**12.321.1.13 averageCol**

```
uint64_t averageCol = 0
```

**12.321.1.14 deviationCol**

```
uint64_t deviationCol = 0
```

**12.321.1.15 minColDifference**

```
uint64_t minColDifference = 0
```

**12.321.1.16 maxColDifference**

```
uint64_t maxColDifference = 0
```

**12.321.1.17 averageColDifference**

```
uint64_t averageColDifference = 0
```

**12.321.1.18 deviationColDifference**

```
uint64_t deviationColDifference = 0
```

**12.321.1.19 minRowDifference**

```
uint64_t minRowDifference = 0
```

**12.321.1.20 maxRowDifference**

```
uint64_t maxRowDifference = 0
```

**12.321.1.21 averageRowDifference**

```
uint64_t averageRowDifference = 0
```

**12.321.1.22 deviationRowDifference**

```
uint64_t deviationRowDifference = 0
```

**12.321.1.23 nDenseRows**

```
uint64_t nDenseRows = 0
```

**12.321.1.24 nDenseCols**

```
uint64_t nDenseCols = 0
```

**12.321.1.25 nEmptyRows**

```
uint64_t nEmptyRows = 0
```

**12.321.1.26 nEmptyCols**

```
uint64_t nEmptyCols = 0
```

**12.321.1.27 nEmptyColsEnd**

```
uint64_t nEmptyColsEnd = 0
```

**12.321.1.28 denseRows**

```
std::vector<uint64_t> denseRows
```

**12.321.1.29 denseCols**

```
std::vector<uint64_t> denseCols
```

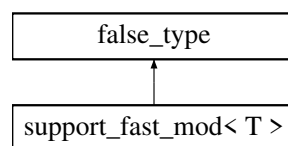
The documentation for this struct was generated from the following file:

- [utils.h](#)

**12.322 support\_fast\_mod< T > Struct Template Reference**

```
#include <fflas_freduce.h>
```

Inheritance diagram for support\_fast\_mod< T >:



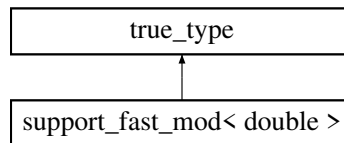
The documentation for this struct was generated from the following file:

- [fflas\\_freduce.h](#)

**12.323 support\_fast\_mod< double > Struct Reference**

```
#include <fflas_freduce.h>
```

Inheritance diagram for support\_fast\_mod< double >:



The documentation for this struct was generated from the following file:

- [fflas\\_freduce.h](#)

### 12.324 support\_fast\_mod< float > Struct Reference

```
#include <fflas_freduce.h>
```

Inheritance diagram for support\_fast\_mod< float >:



The documentation for this struct was generated from the following file:

- [fflas\\_freduce.h](#)

### 12.325 support\_fast\_mod< int64\_t > Struct Reference

```
#include <fflas_freduce.h>
```

Inheritance diagram for support\_fast\_mod< int64\_t >:



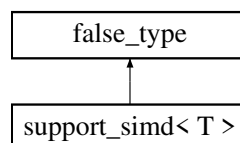
The documentation for this struct was generated from the following file:

- [fflas\\_freduce.h](#)

### 12.326 support\_simd< T > Struct Template Reference

```
#include <fflas_simd.h>
```

Inheritance diagram for support\_simd< T >:



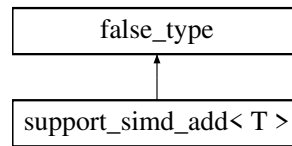
The documentation for this struct was generated from the following file:

- [fflas\\_simd.h](#)

## 12.327 support\_simd\_add< T > Struct Template Reference

```
#include <fflas_fadd.h>
```

Inheritance diagram for support\_simd\_add< T >:



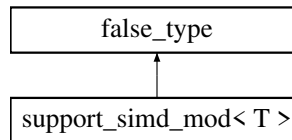
The documentation for this struct was generated from the following file:

- [fflas\\_fadd.h](#)

## 12.328 support\_simd\_mod< T > Struct Template Reference

```
#include <fflas_freduce.h>
```

Inheritance diagram for support\_simd\_mod< T >:



The documentation for this struct was generated from the following file:

- [fflas\\_freduce.h](#)

## 12.329 Test< Elt > Class Template Reference

### Public Types

- using [Field](#) = Modular<Elt>
- using [Elt\\_ptr](#) = typename [Field::Element\\_ptr](#)
- using [Residu](#) = typename [Field::Residu\\_t](#)
- template<bool B, class T = void>  
using [enable\\_if\\_t](#) = typename std::enable\_if<B, T>::type
- template<typename [Simd](#) >  
using [is\\_same\\_element](#) = typename [Simd::template is\\_same\\_element<\[Field\]\(#\)>](#)
- template<typename E >  
using [enable\\_if\\_no\\_simd\\_t](#) = [enable\\_if\\_t](#)<[Simd](#)<E>::vect\_size == 1>
- template<typename E >  
using [enable\\_if\\_simd128\\_t](#) = [enable\\_if\\_t](#)<sizeof(E)\*[Simd](#)<E>::vect\_size == 16>
- template<typename E >  
using [enable\\_if\\_simd256\\_t](#) = [enable\\_if\\_t](#)<sizeof(E)\*[Simd](#)<E>::vect\_size == 32>
- template<typename E >  
using [enable\\_if\\_simd512\\_t](#) = [enable\\_if\\_t](#)<sizeof(E)\*[Simd](#)<E>::vect\_size == 64>

### Public Member Functions

- [Test](#) (size\_t mm, size\_t nn)
- template<typename [Simd](#) = NoSimd<Elt>, [enable\\_if\\_t](#)< [is\\_same\\_element](#)< [Simd](#) >::value > \* = nullptr>  
bool [test\\_franspose](#) (size\_t m, size\_t n, [Elt\\_ptr](#) A, size\_t lda, [Elt\\_ptr](#) B, size\_t ldb)

- `template<typename Simd = NoSimd<Elt>, enable_if_t< is_same_element< Simd >::value > * = nullptr>  
bool doTests ()`
- `template<typename _E = Elt, enable_if_t< is_same< _E, Elt >::value > * = nullptr, enable_if_no_simd_t< _E > * = nullptr>  
bool run ()`

### Static Public Member Functions

- `template<typename _E = Elt, enable_if_t< !is_same< _E, Givaro::Integer >::value > * = nullptr>  
static Residu cardinality ()`
- `template<typename _E = Elt, enable_if_t< is_same< _E, Givaro::Integer >::value > * = nullptr>  
static Residu cardinality ()`

### Protected Attributes

- `Field F`
- `size_t _mm`
- `size_t _nn`

## 12.329.1 Member Typedef Documentation

### 12.329.1.1 Field

```
template<typename Elt >
using Field = Modular<Elt>
```

### 12.329.1.2 Elt\_ptr

```
template<typename Elt >
using Elt_ptr = typename Field::Element_ptr
```

### 12.329.1.3 Residu

```
template<typename Elt >
using Residu = typename Field::Residu_t
```

### 12.329.1.4 enable\_if\_t

```
template<typename Elt >
template<bool B, class T = void>
using enable_if_t = typename std::enable_if<B, T>::type
```

### 12.329.1.5 is\_same\_element

```
template<typename Elt >
template<typename Simd >
using is_same_element = typename Simd::template is_same_element<Field>
```

### 12.329.1.6 enable\_if\_no\_simd\_t

```
template<typename Elt >
template<typename E >
using enable_if_no_simd_t = enable_if_t<Simd<E>::vect_size == 1>
```

### 12.329.1.7 enable\_if\_simd128\_t

```
template<typename Elt >
template<typename E >
using enable_if_simd128_t = enable_if_t<sizeof(E)*Simd<E>::vect_size == 16>
```



**12.329.1.8 enable\_if\_simd256\_t**

```
template<typename Elt >
template<typename E >
using enable_if_simd256_t = enable_if_t<sizeof(E)*Simd<E>::vect_size == 32>
```

**12.329.1.9 enable\_if\_simd512\_t**

```
template<typename Elt >
template<typename E >
using enable_if_simd512_t = enable_if_t<sizeof(E)*Simd<E>::vect_size == 64>
```

**12.329.2 Constructor & Destructor Documentation****12.329.2.1 Test()**

```
template<typename Elt >
Test (
    size_t mm,
    size_t nn) [inline]
```

**12.329.3 Member Function Documentation****12.329.3.1 cardinality() [1/2]**

```
template<typename Elt >
template<typename _E = Elt, enable_if_t<!is_same< _E, Givaro::Integer >::value > * = nullptr>
static Residu cardinality () [inline], [static]
```

**12.329.3.2 cardinality() [2/2]**

```
template<typename Elt >
template<typename _E = Elt, enable_if_t< is_same< _E, Givaro::Integer >::value > * = nullptr>
static Residu cardinality () [inline], [static]
```

**12.329.3.3 test\_ftranspose()**

```
template<typename Elt >
template<typename Simd = NoSimd<Elt>, enable_if_t< is_same_element< Simd >::value > * =
nullptr>
bool test_ftranspose (
    size_t m,
    size_t n,
    Elt_ptr A,
    size_t lda,
    Elt_ptr B,
    size_t ldb) [inline]
```

**12.329.3.4 doTests()**

```
template<typename Elt >
template<typename Simd = NoSimd<Elt>, enable_if_t< is_same_element< Simd >::value > * =
nullptr>
bool doTests () [inline]
```

**12.329.3.5 run()**

```
template<typename Elt >
template<typename _E = Elt, enable_if_t< is_same< _E, Elt >::value > * = nullptr, enable_if_no_simd_t<
_E > * = nullptr>
bool run () [inline]
```

## 12.329.4 Field Documentation

### 12.329.4.1 F

```
template<typename Elt >
Field F [protected]
```

### 12.329.4.2 \_mm

```
template<typename Elt >
size_t _mm [protected]
```

### 12.329.4.3 \_nn

```
template<typename Elt >
size_t _nn [protected]
```

The documentation for this class was generated from the following file:

- [test-storage-transpose.C](#)

## 12.330 TestOneMethod< Simd > Class Template Reference

### Public Types

- using [Element](#) = typename Simd::scalar\_t
- using [vect\\_t](#) = typename Simd::vect\_t
- using [vectElt](#) = vector<[Element](#)>
- template<bool B, typename T = void>  
using [enable\\_if\\_t](#) = typename enable\_if<B, T>::type

### Public Member Functions

- template<typename... AScal, typename RScal, typename... ASimd, typename RSimd, [enable\\_if\\_t](#)< sizeof...(AScal)==sizeof...(ASimd)>  
\* = nullptr, [enable\\_if\\_t](#)< [count\\_nonconst\\_lvalue\\_reference](#)< AScal... >::n==[count\\_nonconst\\_lvalue\\_reference](#)< ASimd... >::n > \*  
= nullptr, [enable\\_if\\_t](#)< [is\\_all\\_same](#)< AScal... >::value > \* = nullptr, [enable\\_if\\_t](#)< [is\\_all\\_same](#)< [vect\\_t](#), ASimd... >::value > \* =  
nullptr>  
[TestOneMethod](#) (function< RSimd(ASimd...)> fsimd, function< RScal(AScal...)> fscal, function<  
void(vector< [vectElt](#) > &)> genInputs, string fname)
- template<typename Ret, typename... AScal>  
[enable\\_if\\_t](#)< [is\\_all\\_same](#)< [Element](#), AScal... >::value &&std::is\_convertible< Ret, [Element](#) >::value, void  
> [evaluate\\_scalar\\_method](#) (function< Ret(AScal...)> fscal)
- template<typename... AScal>  
[enable\\_if\\_t](#)< [is\\_all\\_same](#)< [vectElt](#), AScal... >::value, void > [evaluate\\_scalar\\_method](#) (function<  
[vectElt](#)(AScal...)> fscal)
- template<typename... AScal>  
[enable\\_if\\_t](#)< [is\\_all\\_same](#)< [vectElt](#), AScal... >::value, void > [evaluate\\_scalar\\_method](#) (function<  
void(AScal...)> fscal)
- template<typename Ret, typename... ASimd>  
[enable\\_if\\_t](#)< [is\\_all\\_same](#)< [vect\\_t](#), ASimd... >::value &&std::is\_convertible< Ret, [vect\\_t](#) >::value, void >  
[evaluate\\_simd\\_method](#) (function< Ret(ASimd...)> fsimd, array< [vect\\_t](#), sizeof...(ASimd)> &simd\_in)
- template<typename... ASimd>  
[enable\\_if\\_t](#)< [is\\_all\\_same](#)< [vect\\_t](#), ASimd... >::value, void > [evaluate\\_simd\\_method](#) (function<  
void(ASimd...)> fsimd, array< [vect\\_t](#), sizeof...(ASimd)> &simd\_in)
- bool [getStatus](#) () const
- string [getTestName](#) () const
- bool [writeResultLine](#) () const
- void [writeDebugData](#) () const

### Static Public Attributes

- static constexpr size\_t [vect\\_size](#) = Simd::vect\_size

### Protected Attributes

- size\_t [nb\\_lref](#)
- string [name](#)
- vector< [vectElt](#) > [inputs](#)
- vector< [vectElt](#) > [outputs\\_simd](#)
- vector< [vectElt](#) > [outputs\\_scalar](#)

## 12.330.1 Member Typedef Documentation

### 12.330.1.1 Element

```
template<typename Simd >
using Element = typename Simd::scalar_t
```

### 12.330.1.2 vect\_t

```
template<typename Simd >
using vect_t = typename Simd::vect_t
```

### 12.330.1.3 vectElt

```
template<typename Simd >
using vectElt = vector<Element>
```

### 12.330.1.4 enable\_if\_t

```
template<typename Simd >
template<bool B, typename T = void>
using enable_if_t = typename enable_if<B, T>::type
```

## 12.330.2 Constructor & Destructor Documentation

### 12.330.2.1 TestOneMethod()

```
template<typename Simd >
template<typename... AScal, typename RScal, typename... ASimd, typename RSimd, enable_if_t<
sizeof...(AScal)==sizeof...(ASimd)> * = nullptr, enable_if_t< count_nonconst_lvalue_reference<
AScal... >::n==count_nonconst_lvalue_reference< ASimd... >::n > * = nullptr, enable_if_t<
is_all_same< AScal... >::value > * = nullptr, enable_if_t< is_all_same< vect_t, ASimd...
>::value > * = nullptr>
TestOneMethod (
    function< RSimd(ASimd...)> fsimd,
    function< RScal(AScal...)> fscal,
    function< void(vector< vectElt > &)> genInputs,
    string fname) [inline]
```

## 12.330.3 Member Function Documentation

### 12.330.3.1 evaluate\_scalar\_method() [1/3]

```
template<typename Simd >
template<typename Ret, typename... AScal>
enable_if_t< is_all_same< Element, AScal... >::value &&std::is_convertible< Ret, Element > &↔
::value, void > evaluate_scalar_method (
    function< Ret(AScal...)> fscal) [inline]
```

**12.330.3.2 evaluate\_scalar\_method() [2/3]**

```
template<typename Simd >
template<typename... AScal>
enable_if_t< is_all_same< vectElt, AScal... >::value, void > evaluate_scalar_method (
    function< vectElt (AScal...)> fscal) [inline]
```

**12.330.3.3 evaluate\_scalar\_method() [3/3]**

```
template<typename Simd >
template<typename... AScal>
enable_if_t< is_all_same< vectElt, AScal... >::value, void > evaluate_scalar_method (
    function< void (AScal...)> fscal) [inline]
```

**12.330.3.4 evaluate\_simd\_method() [1/2]**

```
template<typename Simd >
template<typename Ret , typename... ASimd>
enable_if_t< is_all_same< vect_t, ASimd... >::value &&std::is_convertible< Ret, vect_t >↔
::value, void > evaluate_simd_method (
    function< Ret (ASimd...)> fsimd,
    array< vect_t, sizeof...(ASimd)> & simd_in) [inline]
```

**12.330.3.5 evaluate\_simd\_method() [2/2]**

```
template<typename Simd >
template<typename... ASimd>
enable_if_t< is_all_same< vect_t, ASimd... >::value, void > evaluate_simd_method (
    function< void (ASimd...)> fsimd,
    array< vect_t, sizeof...(ASimd)> & simd_in) [inline]
```

**12.330.3.6 getStatus()**

```
template<typename Simd >
bool getStatus () const [inline]
```

**12.330.3.7 getTestName()**

```
template<typename Simd >
string getTestName () const [inline]
```

**12.330.3.8 writeResultLine()**

```
template<typename Simd >
bool writeResultLine () const [inline]
```

**12.330.3.9 writeDebugData()**

```
template<typename Simd >
void writeDebugData () const [inline]
```

**12.330.4 Field Documentation****12.330.4.1 vect\_size**

```
template<typename Simd >
size_t vect_size = Simd::vect_size [static], [constexpr]
```

**12.330.4.2 nb\_lref**

```
template<typename Simd >
size_t nb_lref [protected]
```

**12.330.4.3 name**

```
template<typename Simd >
string name [protected]
```

**12.330.4.4 inputs**

```
template<typename Simd >
vector<vectElt> inputs [protected]
```

**12.330.4.5 outputs\_simd**

```
template<typename Simd >
vector<vectElt> outputs_simd [protected]
```

**12.330.4.6 outputs\_scalar**

```
template<typename Simd >
vector<vectElt> outputs_scalar [protected]
```

The documentation for this class was generated from the following file:

- [test-simd.C](#)

**12.331 tfn\_minus Struct Reference**

```
#include <sparse_matrix_traits.h>
```

**Public Member Functions**

- `template<typename... Args>`  
`auto operator() (Args &&... args) const -> decltype(operator+(std::forward< Args >(args)...))`

**12.331.1 Member Function Documentation****12.331.1.1 operator>()()**

```
template<typename... Args>
auto operator() (
    Args &&... args) const -> decltype(operator+(std::forward<Args>(args)...))

[inline]
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

**12.332 tfn\_minus\_eq Struct Reference**

```
#include <sparse_matrix_traits.h>
```

**Public Member Functions**

- `template<typename... Args>`  
`auto operator() (Args &&... args) const -> decltype(operator+(std::forward< Args >(args)...))`

## 12.332.1 Member Function Documentation

### 12.332.1.1 operator>()

```
template<typename... Args>
auto operator() (
    Args &&... args) const -> decltype(operator+(std::forward<Args>(args)...))
[inline]
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.333 tfn\_mul Struct Reference

```
#include <sparse_matrix_traits.h>
```

### Public Member Functions

- template<typename... Args>  
auto [operator\(\)](#) (Args &&... args) const -> decltype(operator+(std::forward< Args >(args)...))

## 12.333.1 Member Function Documentation

### 12.333.1.1 operator>()

```
template<typename... Args>
auto operator() (
    Args &&... args) const -> decltype(operator+(std::forward<Args>(args)...))
[inline]
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.334 tfn\_mul\_eq Struct Reference

```
#include <sparse_matrix_traits.h>
```

### Public Member Functions

- template<typename... Args>  
auto [operator\(\)](#) (Args &&... args) const -> decltype(operator+(std::forward< Args >(args)...))

## 12.334.1 Member Function Documentation

### 12.334.1.1 operator>()

```
template<typename... Args>
auto operator() (
    Args &&... args) const -> decltype(operator+(std::forward<Args>(args)...))
[inline]
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.335 tfn\_plus Struct Reference

```
#include <sparse_matrix_traits.h>
```

**Public Member Functions**

- `template<typename... Args>`  
`auto operator() (Args &&... args) const -> decltype(operator+(std::forward< Args >(args)...))`

**12.335.1 Member Function Documentation****12.335.1.1 operator>()()**

```
template<typename... Args>
auto operator() (
    Args &&... args) const -> decltype(operator+(std::forward<Args>(args)...))
[inline]
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

**12.336 tfn\_plus\_eq Struct Reference**

```
#include <sparse_matrix_traits.h>
```

**Public Member Functions**

- `template<typename... Args>`  
`auto operator() (Args &&... args) const -> decltype(operator+(std::forward< Args >(args)...))`

**12.336.1 Member Function Documentation****12.336.1.1 operator>()()**

```
template<typename... Args>
auto operator() (
    Args &&... args) const -> decltype(operator+(std::forward<Args>(args)...))
[inline]
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

**12.337 Threads Struct Reference**

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

**12.338 ThreeD Struct Reference**

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

**12.339 ThreeDAdaptive Struct Reference**

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.340 ThreeDInPlace Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.341 TRSMHelper< RecIterTrait, ParSeqTrait > Struct Template Reference

TRSM Helper.

### Public Member Functions

- `template<class Cut , class Param >`  
[TRSMHelper](#) ([ParSeqHelper::Parallel](#)< Cut, Param > \_PS)
- `TRSMHelper` ([ParSeqHelper::Sequential](#) \_PS)
- `template<typename RIT , typename PST >`  
[TRSMHelper](#) ([TRSMHelper](#)< RIT, PST > &\_TH)
- `template<class Dom , class Algo = FFLAS::MMHelperAlgo::Winograd, class ModeT = typename FFLAS::ModeTraits<Dom>::value>`  
[FFLAS::MMHelper](#)< Dom, Algo, ModeT, ParSeqTrait > [pMMH](#) (Dom &D, size\_t m, size\_t k, size\_t n, ParSeqTrait p) const
- `template<class Dom , class Algo = FFLAS::MMHelperAlgo::Winograd, class ModeT = typename FFLAS::ModeTraits<Dom>::value>`  
[FFLAS::MMHelper](#)< Dom, Algo, ModeT, ParSeqTrait > [pMMH](#) (Dom &D, size\_t m, size\_t k, size\_t n) const

### Data Fields

- ParSeqTrait [parseq](#)

### 12.341.1 Detailed Description

```
template<typename RecIterTrait = StructureHelper::Recursive, typename ParSeqTrait = ParSeqHelper::Sequential>
struct FFLAS::TRSMHelper< RecIterTrait, ParSeqTrait >
```

TRSM Helper.

### 12.341.2 Constructor & Destructor Documentation

#### 12.341.2.1 TRSMHelper() [1/3]

```
template<typename RecIterTrait = StructureHelper::Recursive, typename ParSeqTrait = ParSeqHelper::Sequential>
template<class Cut , class Param >
TRSMHelper (
    ParSeqHelper::Parallel< Cut, Param > _PS) [inline]
```

#### 12.341.2.2 TRSMHelper() [2/3]

```
template<typename RecIterTrait = StructureHelper::Recursive, typename ParSeqTrait = ParSeqHelper::Sequential>
TRSMHelper (
    ParSeqHelper::Sequential _PS) [inline]
```

#### 12.341.2.3 TRSMHelper() [3/3]

```
template<typename RecIterTrait = StructureHelper::Recursive, typename ParSeqTrait = ParSeqHelper::Sequential>
template<typename RIT , typename PST >
TRSMHelper (
    TRSMHelper< RIT, PST > &_TH) [inline]
```



### 12.341.3 Member Function Documentation

#### 12.341.3.1 pMMH() [1/2]

```
template<typename RecIterTrait = StructureHelper::Recursive, typename ParSeqTrait = ParSeq↵
Helper::Sequential>
template<class Dom , class Algo = FFLAS::MMHelperAlgo::Winograd, class ModeT = typename FFLAS↵
::ModeTraits<Dom>::value>
FFLAS::MMHelper< Dom, Algo, ModeT, ParSeqTrait > pMMH (
    Dom & D,
    size_t m,
    size_t k,
    size_t n,
    ParSeqTrait p) const [inline]
```

#### 12.341.3.2 pMMH() [2/2]

```
template<typename RecIterTrait = StructureHelper::Recursive, typename ParSeqTrait = ParSeq↵
Helper::Sequential>
template<class Dom , class Algo = FFLAS::MMHelperAlgo::Winograd, class ModeT = typename FFLAS↵
::ModeTraits<Dom>::value>
FFLAS::MMHelper< Dom, Algo, ModeT, ParSeqTrait > pMMH (
    Dom & D,
    size_t m,
    size_t k,
    size_t n) const [inline]
```

### 12.341.4 Field Documentation

#### 12.341.4.1 parseq

```
template<typename RecIterTrait = StructureHelper::Recursive, typename ParSeqTrait = ParSeq↵
Helper::Sequential>
ParSeqTrait parseq
```

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

## 12.342 TwoD Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.343 TwoDAdaptive Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.344 UnparametricTag Struct Reference

If the field uses a representation with infix operators.

```
#include <field-traits.h>
```

### 12.344.1 Detailed Description

If the field uses a representation with infix operators.

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.345 `width< T >` Struct Template Reference

### Static Public Attributes

- static constexpr `size_t value` = 2+2\*sizeof(T)

### 12.345.1 Field Documentation

#### 12.345.1.1 `value`

```
template<typename T >
size_t value = 2+2*sizeof(T) [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.346 `width< double >` Struct Reference

### Static Public Attributes

- static constexpr `size_t value` = 24

### 12.346.1 Field Documentation

#### 12.346.1.1 `value`

```
size_t value = 24 [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.347 `width< float >` Struct Reference

### Static Public Attributes

- static constexpr `size_t value` = 16

### 12.347.1 Field Documentation

#### 12.347.1.1 `value`

```
size_t value = 16 [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.348 Winograd Struct Reference

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

## 12.349 WinogradPar Struct Reference

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

# Chapter 13

## File Documentation

### 13.1 arithprog.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
```

#### Typedefs

- typedef Givaro::OMPTimer [TTimer](#)

#### Functions

- int [main](#) (int argc, char \*\*argv)

#### 13.1.1 Typedef Documentation

##### 13.1.1.1 TTimer

```
typedef FFLAS::Timer TTimer
```

#### 13.1.2 Function Documentation

##### 13.1.2.1 main()

```
int main (
    int argc,
    char ** argv)
```

### 13.2 autotune/charpoly.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include <iostream>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include <ctime>
```

#### Macros

- #define [CUBE](#)(x)
- #define [GFOPS](#)(m, n, r, t)

## Typedefs

- typedef Givaro::Timer [TTimer](#)

## Functions

- int [main](#) ()

### 13.2.1 Macro Definition Documentation

#### 13.2.1.1 CUBE

```
#define CUBE(
                x)
```

##### Value:

```
((x)*(x)*(x))
```

#### 13.2.1.2 GFOPS

```
#define GFOPS(
                m,
                n,
                r,
                t)
```

##### Value:

```
(2.7*CUBE(double(n)/1000.0))/t
```

### 13.2.2 Typedef Documentation

#### 13.2.2.1 TTimer

```
typedef Givaro::Timer TTimer
```

### 13.2.3 Function Documentation

#### 13.2.3.1 main()

```
int main (
        void )
```

## 13.3 examples/charpoly.C File Reference

```
#include <iostream>
#include "fflas-ffpack/fflas-ffpack.h"
```

## Functions

- int [main](#) (int argc, char \*\*argv)

*This example computes the characteristic polynomial of a matrix over a defined finite field.*

### 13.3.1 Function Documentation

#### 13.3.1.1 main()

```
int main (
        int argc,
        char ** argv)
```

This example computes the characteristic polynomial of a matrix over a defined finite field.  
Outputs the characteristic polynomial.

## 13.4 fsyrk.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include <ctime>
```

### Macros

- #define CUBE(x)
- #define GFOPS(n, t)

### Functions

- int main ()

### 13.4.1 Macro Definition Documentation

#### 13.4.1.1 CUBE

```
#define CUBE(  
    x)
```

##### Value:

```
((x) * (x) * (x))
```

#### 13.4.1.2 GFOPS

```
#define GFOPS(  
    n,  
    t)
```

##### Value:

```
(CUBE(double(n) / 1000.0) / (3.0 * t))
```

### 13.4.2 Function Documentation

#### 13.4.2.1 main()

```
int main (  
    void )
```

## 13.5 fsytrf.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include <ctime>
```

### Macros

- #define CUBE(x)
- #define GFOPS(n, t)

## Functions

- int [main](#) ()

## 13.5.1 Macro Definition Documentation

### 13.5.1.1 CUBE

```
#define CUBE(  
                x)
```

#### Value:

```
((x) * (x) * (x))
```

### 13.5.1.2 GFOPS

```
#define GFOPS(  
                n,  
                t)
```

#### Value:

```
(CUBE(double(n) / 1000.0) / (3.0 * t))
```

## 13.5.2 Function Documentation

### 13.5.2.1 main()

```
int main (  
            void )
```

## 13.6 ftrtri.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"  
#include "fflas-ffpack/utils/fflas_randommatrix.h"  
#include <iostream>  
#include <givaro/modular-balanced.h>  
#include "fflas-ffpack/utils/timer.h"  
#include "fflas-ffpack/ffpack/ffpack.h"  
#include <ctime>
```

## Macros

- #define [CUBE](#)(x)
- #define [GFOPS](#)(n, t)

## Functions

- int [main](#) ()

## 13.6.1 Macro Definition Documentation

### 13.6.1.1 CUBE

```
#define CUBE(  
                x)
```

#### Value:

```
((x) * (x) * (x))
```

### 13.6.1.2 GFOPS

```
#define GFOPS(
    n,
    t)
```

**Value:**

```
(CUBE(double(n)/1000.0)/(3.0*t))
```

## 13.6.2 Function Documentation

### 13.6.2.1 main()

```
int main (
    void )
```

## 13.7 autotune/pluq.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include <iostream>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include <ctime>
```

### Macros

- #define CUBE(x)
- #define GFOPS(m, n, r, t)

### Functions

- int main ()

## 13.7.1 Macro Definition Documentation

### 13.7.1.1 CUBE

```
#define CUBE(
    x)
```

**Value:**

```
((x)*(x)*(x))
```

### 13.7.1.2 GFOPS

```
#define GFOPS(
    m,
    n,
    r,
    t)
```

**Value:**

```
(2.0/3.0*CUBE(double(n)/1000.0) + 2*m/1000.0*n/1000.0*double(r)/1000.0 -
double(r)/1000.0*double(r)/1000.0*(m+n)/1000)/t
```

## 13.7.2 Function Documentation

### 13.7.2.1 main()

```
int main (
    void )
```

## 13.8 examples/pluq.C File Reference

```
#include <iostream>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/fflas_io.h"
```

### Functions

- int [main](#) (int argc, char \*\*argv)

### 13.8.1 Function Documentation

#### 13.8.1.1 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.9 winograd.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include <iostream>
#include <fstream>
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include <ctime>
```

### Macros

- #define [DOUBLE\\_TO\\_FLOAT\\_CROSSOVER](#) 0
- #define [GFOPS](#)(n, t)

### Functions

- template<class [Field](#) >  
bool [balanced](#) (const [Field](#) &)
- template<class T >  
bool [balanced](#) (const [Givaro::ModularBalanced](#)< T > &)
- int [main](#) ()

### 13.9.1 Macro Definition Documentation

#### 13.9.1.1 DOUBLE\_TO\_FLOAT\_CROSSOVER

```
#define DOUBLE_TO_FLOAT_CROSSOVER 0
```

#### 13.9.1.2 GFOPS

```
#define GFOPS(
    n,
    t)
```

#### Value:

```
(2.0/t*(double)n/1000.0*(double)n/1000.0*(double)n/1000.0)
```



## 13.9.2 Function Documentation

### 13.9.2.1 `balanced()` [1/2]

```
template<class Field >
bool balanced (
    const Field & )
```

### 13.9.2.2 `balanced()` [2/2]

```
template<class T >
bool balanced (
    const Givaro::ModularBalanced< T > & )
```

### 13.9.2.3 `main()`

```
int main (
    void )
```

## 13.10 benchmark-charpoly-mp.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/Matio.h"
#include "fflas-ffpack/utils/args-parser.h"
```

### Macros

- `#define __FFLASFFPACK_FORCE_SEQ`

### Functions

- `int main` (int argc, char \*\*argv)

## 13.10.1 Macro Definition Documentation

### 13.10.1.1 `__FFLASFFPACK_FORCE_SEQ`

```
#define __FFLASFFPACK_FORCE_SEQ
```

## 13.10.2 Function Documentation

### 13.10.2.1 `main()`

```
int main (
    int argc,
    char ** argv)
```

## 13.11 benchmark-charpoly.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include <givaro/givpoly1.h>
```

```
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

## Macros

- `#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1`

## Functions

- `template<class Field >`  
`void run_with_field (int q, uint64_t bits, size_t n, size_t d, size_t iter, std::string file, int variant, uint64_t seed)`
- `int main (int argc, char **argv)`

## 13.11.1 Macro Definition Documentation

### 13.11.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

## 13.11.2 Function Documentation

### 13.11.2.1 run\_with\_field()

```
template<class Field >
void run_with_field (
    int q,
    uint64_t bits,
    size_t n,
    size_t d,
    size_t iter,
    std::string file,
    int variant,
    uint64_t seed)
```

### 13.11.2.2 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.12 benchmark-checkers.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/checkers/checkers_fflas.h"
#include "fflas-ffpack/checkers/checkers_ffpack.h"
```

```
#include <fstream>
```

### Macros

- #define [ENABLE\\_ALL\\_CHECKINGS](#) 1
- #define [\\_NR\\_TESTS](#) 5
- #define [\\_MAX\\_SIZE\\_MATRICES](#) 1000
- #define [CUBE](#)(x)

### Functions

- int [main](#) (int argc, char \*\*argv)

## 13.12.1 Macro Definition Documentation

### 13.12.1.1 [ENABLE\\_ALL\\_CHECKINGS](#)

```
#define ENABLE_ALL_CHECKINGS 1
```

### 13.12.1.2 [\\_NR\\_TESTS](#)

```
#define _NR_TESTS 5
```

### 13.12.1.3 [\\_MAX\\_SIZE\\_MATRICES](#)

```
#define _MAX_SIZE_MATRICES 1000
```

### 13.12.1.4 [CUBE](#)

```
#define CUBE(  
    x)
```

#### Value:

```
((x) * (x) * (x))
```

## 13.12.2 Function Documentation

### 13.12.2.1 [main\(\)](#)

```
int main (  
    int argc,  
    char ** argv)
```

## 13.13 benchmark-dgemm.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"  
#include <iostream>  
#include <givaro/modular.h>  
#include "fflas-ffpack/config-blas.h"  
#include "fflas-ffpack/fflas/fflas.h"  
#include "fflas-ffpack/utils/timer.h"  
#include "fflas-ffpack/utils/fflas_io.h"  
#include "fflas-ffpack/utils/args-parser.h"
```

### Macros

- #define [CBLAS\\_GEMM](#) cblas\_dgemm

**Typedefs**

- typedef [FFLAS::Timer](#) TTimer
- typedef double [Floats](#)

**Functions**

- int [main](#) (int argc, char \*\*argv)

**13.13.1 Macro Definition Documentation****13.13.1.1 CBLAS\_GEMM**

```
#define CBLAS_GEMM cblas_dgemm
```

**13.13.2 Typedef Documentation****13.13.2.1 TTimer**

```
typedef FFLAS::Timer TTimer
```

**13.13.2.2 Floats**

```
typedef double Floats
```

**13.13.3 Function Documentation****13.13.3.1 main()**

```
int main (
    int argc,
    char ** argv)
```

**13.14 benchmark-dgetrf.C File Reference**

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <vector>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

**Macros**

- #define [\\_\\_FFLASFFPACK\\_HAVE\\_DGETRF](#) 1

**Functions**

- int [main](#) (int argc, char \*\*argv)

**13.14.1 Macro Definition Documentation****13.14.1.1 \_\_FFLASFFPACK\_HAVE\_DGETRF**

```
#define \_\_FFLASFFPACK\_HAVE\_DGETRF 1
```

## 13.14.2 Function Documentation

### 13.14.2.1 main()

```
int main (  
    int argc,  
    char ** argv)
```

## 13.15 benchmark-dgetri.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"  
#include <iostream>  
#include <vector>  
#include <givaro/modular.h>  
#include "fflas-ffpack/fflas-ffpack.h"  
#include "fflas-ffpack/utils/timer.h"  
#include "fflas-ffpack/utils/fflas_io.h"  
#include "fflas-ffpack/utils/args-parser.h"
```

### Functions

- int [main](#) (int argc, char \*\*argv)

## 13.15.1 Function Documentation

### 13.15.1.1 main()

```
int main (  
    int argc,  
    char ** argv)
```

## 13.16 benchmark-dsytrf.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"  
#include <iostream>  
#include <vector>  
#include <givaro/modular.h>  
#include "fflas-ffpack/fflas-ffpack.h"  
#include "fflas-ffpack/utils/timer.h"  
#include "fflas-ffpack/utils/fflas_io.h"  
#include "fflas-ffpack/utils/args-parser.h"
```

### Macros

- #define [EFFGFF](#)(n, t, i)

### Functions

- int [main](#) (int argc, char \*\*argv)

## 13.16.1 Macro Definition Documentation

### 13.16.1.1 EFFGFF

```
#define EFFGFF(  
    n,
```

```
t,
i)
```

**Value:**

```
( (double(n)/1000.*double(n)/1000.*double(n)/1000.0) / double(t) * double(i) / 3.)
```

**13.16.2 Function Documentation****13.16.2.1 main()**

```
int main (
    int argc,
    char ** argv)
```

**13.17 benchmark-dtrsm.C File Reference**

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

**Functions**

- int [main](#) (int argc, char \*\*argv)

**13.17.1 Function Documentation****13.17.1.1 main()**

```
int main (
    int argc,
    char ** argv)
```

**13.18 benchmark-dtrtri.C File Reference**

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

**Macros**

- #define [\\_\\_FFLASFFPACK\\_HAVE\\_DTRTRI](#) 1

**Functions**

- int [main](#) (int argc, char \*\*argv)

### 13.18.1 Macro Definition Documentation

#### 13.18.1.1 \_\_FFLASFFPACK\_HAVE\_DTRTRI

```
#define __FFLASFFPACK_HAVE_DTRTRI 1
```

### 13.18.2 Function Documentation

#### 13.18.2.1 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.19 benchmark-fadd-lvl2.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

### Macros

- `#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1`

### Functions

- `int main (int argc, char **argv)`

### 13.19.1 Macro Definition Documentation

#### 13.19.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

### 13.19.2 Function Documentation

#### 13.19.2.1 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.20 benchmark-fdot.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include <givaro/givrational.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/paladin/parallel.h"
#include "fflas-ffpack/paladin/fflas_plevel1.h"
```

```
#include "fflas-ffpack/utils/args-parser.h"
```

## Macros

- `#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1`

## Functions

- `template<class Field >`  
`Field::Element run_with_field` (int q, size\_t iter, size\_t N, const uint64\_t BS, const size\_t p, const size\_t threads, uint64\_t seed)
- `int main` (int argc, char \*\*argv)

## 13.20.1 Macro Definition Documentation

### 13.20.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

## 13.20.2 Function Documentation

### 13.20.2.1 run\_with\_field()

```
template<class Field >
Field::Element run_with_field (
    int q,
    size_t iter,
    size_t N,
    const uint64_t BS,
    const size_t p,
    const size_t threads,
    uint64_t seed)
```

### 13.20.2.2 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.21 benchmark-fgemm-mp.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <typeinfo>
#include <vector>
#include <string>
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "givaro/modular-integer.h"
#include "givaro/givcaster.h"
#include "fflas-ffpack/paladin/parallel.h"
```



## Macros

- `#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1`
- `#define MG_DEFAULT MG_ACTIVE`
- `#define STD_RECINT_SIZE 8`

## Functions

- `template<typename Ints >  
int tmain ()`
- `int main (int argc, char **argv)`

### 13.21.1 Macro Definition Documentation

#### 13.21.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

#### 13.21.1.2 MG\_DEFAULT

```
#define MG_DEFAULT MG_ACTIVE
```

#### 13.21.1.3 STD\_RECINT\_SIZE

```
#define STD_RECINT_SIZE 8
```

### 13.21.2 Function Documentation

#### 13.21.2.1 tmain()

```
template<typename Ints >  
int tmain ()
```

#### 13.21.2.2 main()

```
int main (  
    int argc,  
    char ** argv)
```

## 13.22 benchmark-fgemm-rns.C File Reference

```
#include "fflas-ffpack/fflas/fflas.h"  
#include <iostream>  
#include "fflas-ffpack/utils/timer.h"  
#include "fflas-ffpack/utils/args-parser.h"
```

## Macros

- `#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1`

## Typedefs

- `typedef FFPACK::rns_double RNS`
- `typedef FFPACK::RNSInteger< RNS > Field`
- `typedef Field::Element_ptr Element_ptr`
- `typedef Field::ConstElement_ptr ConstElement_ptr`
- `typedef StrategyParameter::Threads THREADS`
- `typedef StrategyParameter::Grain GRAIN`

- typedef [StrategyParameter::TwoD](#) TWOD
- typedef [StrategyParameter::TwoDAdaptive](#) TWODA
- typedef [StrategyParameter::ThreeD](#) THREED
- typedef [StrategyParameter::ThreeDAdaptive](#) THREEDA
- typedef [StrategyParameter::ThreeDInPlace](#) THREEDIP
- typedef [ParSeqHelper::Sequential](#) PSeq

## Functions

- int [main](#) (int argc, char \*argv[])

## 13.22.1 Macro Definition Documentation

### 13.22.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

## 13.22.2 Typedef Documentation

### 13.22.2.1 RNS

```
typedef FFPACK::rns\_double RNS
```

### 13.22.2.2 Field

```
typedef FFPACK::RNSInteger<RNS> Field
```

### 13.22.2.3 Element\_ptr

```
typedef Field::Element\_ptr Element_ptr
```

### 13.22.2.4 ConstElement\_ptr

```
typedef Field::ConstElement\_ptr ConstElement_ptr
```

### 13.22.2.5 THREADS

```
typedef StrategyParameter::Threads THREADS
```

### 13.22.2.6 GRAIN

```
typedef StrategyParameter::Grain GRAIN
```

### 13.22.2.7 TWOD

```
typedef StrategyParameter::TwoD TWOD
```

### 13.22.2.8 TWODA

```
typedef StrategyParameter::TwoDAdaptive TWODA
```

### 13.22.2.9 THREED

```
typedef StrategyParameter::ThreeD THREED
```

### 13.22.2.10 THREEDA

```
typedef StrategyParameter::ThreeDAdaptive THREEDA
```

**13.22.2.11 THREEDIP**

```
typedef StrategyParameter::ThreeDInPlace THREEDIP
```

**13.22.2.12 PSeq**

```
typedef ParSeqHelper::Sequential PSeq
```

**13.22.3 Function Documentation****13.22.3.1 main()**

```
int main (
    int argc,
    char * argv[])
```

**13.23 benchmark-fgemm.C File Reference**

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/config-blas.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/args-parser.h"
```

**Macros**

- #define CLASSIC\_HYBRID

**Functions**

- int main (int argc, char \*\*argv)

**13.23.1 Macro Definition Documentation****13.23.1.1 CLASSIC\_HYBRID**

```
#define CLASSIC_HYBRID
```

**13.23.2 Function Documentation****13.23.2.1 main()**

```
int main (
    int argc,
    char ** argv)
```

**13.24 benchmark-fgemv-mp.C File Reference**

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <typeinfo>
#include <vector>
#include <string>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
```

```
#include "givaro/modular-integer.h"
#include "givaro/givcaster.h"
#include "fflas-ffpack/paladin/parallel.h"
```

## Macros

- `#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1`
- `#define MG_DEFAULT MG_ACTIVE`
- `#define STD_RECINT_SIZE 8`

## Functions

- `template<typename T >`  
`std::ostream & write_matrix (std::ostream &out, Givaro::Integer p, size_t m, size_t n, T *C, size_t ldc)`
- `template<typename Ints >`  
`int tmain ()`
- `int main (int argc, char **argv)`

## 13.24.1 Macro Definition Documentation

### 13.24.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

### 13.24.1.2 MG\_DEFAULT

```
#define MG_DEFAULT MG_ACTIVE
```

### 13.24.1.3 STD\_RECINT\_SIZE

```
#define STD_RECINT_SIZE 8
```

## 13.24.2 Function Documentation

### 13.24.2.1 write\_matrix()

```
template<typename T >
std::ostream & write_matrix (
    std::ostream & out,
    Givaro::Integer p,
    size_t m,
    size_t n,
    T * C,
    size_t ldc)
```

### 13.24.2.2 tmain()

```
template<typename Ints >
int tmain ()
```

### 13.24.2.3 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.25 benchmark-fgemv.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/config-blas.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "givaro/modular-integer.h"
#include "givaro/givcaster.h"
```

### Data Structures

- struct [need\\_field\\_characteristic](#)< Field >
- struct [need\\_field\\_characteristic](#)< Givaro::Modular< Field > >
- struct [need\\_field\\_characteristic](#)< Givaro::ModularBalanced< Field > >
- struct [compatible\\_data\\_type](#)< Field >
- struct [compatible\\_data\\_type](#)< Givaro::ZRing< float > >
- struct [compatible\\_data\\_type](#)< Givaro::ZRing< double > >

### Macros

- #define [\\_\\_FFLASFFPACK\\_OPENBLAS\\_NT\\_ALREADY\\_SET](#) 1

### Functions

- template<class [Field](#) , class RandIter , class Matrix , class Vector >  
void [fill\\_value](#) ([Field](#) &F, RandIter &Rand, Matrix &A, Vector &X, Vector &Y, size\_t m, size\_t k, size\_t incX, size\_t incY, size\_t lda, int NBK)
- template<class [Field](#) , class Matrix , class Vector >  
void [genData](#) ([Field](#) &F, Matrix &A, Vector &X, Vector &Y, size\_t m, size\_t k, size\_t incX, size\_t incY, size\_t lda, int NBK, uint64\_t bitsize, uint64\_t seed)
- template<class [Field](#) , class Matrix , class Vector >  
bool [check\\_result](#) ([Field](#) &F, size\_t m, size\_t lda, Matrix &A, Vector &X, size\_t incX, Vector &Y, size\_t incY)
- template<class [Field](#) , class Matrix , class Vector >  
bool [benchmark\\_with\\_timer](#) ([Field](#) &F, int p, Matrix &A, Vector &X, Vector &Y, size\_t m, size\_t k, size\_t incX, size\_t incY, size\_t lda, size\_t iters, int t, double &time, size\_t GrainSize)
- template<class [Field](#) , class arg >  
void [benchmark\\_disp](#) ([Field](#) &F, bool pass, double &time, size\_t iters, int p, size\_t m, size\_t k, arg &as)
- template<class [Field](#) , class arg >  
void [benchmark\\_in\\_Field](#) ([Field](#) &F, int p, size\_t m, size\_t k, int NBK, uint64\_t bitsize, uint64\_t seed, size\_t iters, int t, arg &as, size\_t GrainSize)
- template<class [Field](#) , class arg >  
void [benchmark\\_with\\_field](#) (int p, size\_t m, size\_t k, int NBK, uint64\_t bitsize, uint64\_t seed, size\_t iters, int t, arg &as, size\_t GrainSize)
- template<class [Field](#) , class arg >  
void [benchmark\\_with\\_field](#) (const Givaro::Integer &q, int p, size\_t m, size\_t k, int NBK, uint64\_t bitsize, uint64\_t seed, size\_t iters, int t, arg &as, size\_t GrainSize)
- int [main](#) (int argc, char \*\*argv)

### 13.25.1 Macro Definition Documentation

#### 13.25.1.1 [\\_\\_FFLASFFPACK\\_OPENBLAS\\_NT\\_ALREADY\\_SET](#)

```
#define \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET 1
```

## 13.25.2 Function Documentation

### 13.25.2.1 fill\_value()

```
template<class Field , class RandIter , class Matrix , class Vector >
void fill_value (
    Field & F,
    RandIter & Rand,
    Matrix & A,
    Vector & X,
    Vector & Y,
    size_t m,
    size_t k,
    size_t incX,
    size_t incY,
    size_t lda,
    int NBK)
```

### 13.25.2.2 genData()

```
template<class Field , class Matrix , class Vector >
void genData (
    Field & F,
    Matrix & A,
    Vector & X,
    Vector & Y,
    size_t m,
    size_t k,
    size_t incX,
    size_t incY,
    size_t lda,
    int NBK,
    uint64_t bitsize,
    uint64_t seed)
```

### 13.25.2.3 check\_result()

```
template<class Field , class Matrix , class Vector >
bool check_result (
    Field & F,
    size_t m,
    size_t lda,
    Matrix & A,
    Vector & X,
    size_t incX,
    Vector & Y,
    size_t incY)
```

### 13.25.2.4 benchmark\_with\_timer()

```
template<class Field , class Matrix , class Vector >
bool benchmark_with_timer (
    Field & F,
    int p,
    Matrix & A,
    Vector & X,
    Vector & Y,
    size_t m,
    size_t k,
    size_t incX,
```

```

    size_t incY,
    size_t lda,
    size_t iters,
    int t,
    double & time,
    size_t GrainSize)

```

#### 13.25.2.5 benchmark\_disp()

```

template<class Field , class arg >
void benchmark_disp (
    Field & F,
    bool pass,
    double & time,
    size_t iters,
    int p,
    size_t m,
    size_t k,
    arg & as)

```

#### 13.25.2.6 benchmark\_in\_Field()

```

template<class Field , class arg >
void benchmark_in_Field (
    Field & F,
    int p,
    size_t m,
    size_t k,
    int NBK,
    uint64_t bitsize,
    uint64_t seed,
    size_t iters,
    int t,
    arg & as,
    size_t GrainSize)

```

#### 13.25.2.7 benchmark\_with\_field() [1/2]

```

template<class Field , class arg >
void benchmark_with_field (
    int p,
    size_t m,
    size_t k,
    int NBK,
    uint64_t bitsize,
    uint64_t seed,
    size_t iters,
    int t,
    arg & as,
    size_t GrainSize)

```

#### 13.25.2.8 benchmark\_with\_field() [2/2]

```

template<class Field , class arg >
void benchmark_with_field (
    const Givaro::Integer & q,
    int p,
    size_t m,
    size_t k,

```

```

    int NBK,
    uint64_t bitsize,
    uint64_t seed,
    size_t iters,
    int t,
    arg & as,
    size_t GrainSize)

```

### 13.25.2.9 main()

```

int main (
    int argc,
    char ** argv)

```

## 13.26 benchmark-fgesv.C File Reference

```

#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"

```

### Macros

- `#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1`

### Functions

- int `main` (int argc, char \*\*argv)

## 13.26.1 Macro Definition Documentation

### 13.26.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```

#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1

```

## 13.26.2 Function Documentation

### 13.26.2.1 main()

```

int main (
    int argc,
    char ** argv)

```

## 13.27 benchmark-fsyr2k.C File Reference

```

#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/config-blas.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/args-parser.h"

```



## Functions

- int [main](#) (int argc, char \*\*argv)

### 13.27.1 Function Documentation

#### 13.27.1.1 main()

```
int main (  
    int argc,  
    char ** argv)
```

## 13.28 benchmark-fsyrrk.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"  
#include <iostream>  
#include <givaro/modular.h>  
#include "fflas-ffpack/fflas-ffpack.h"  
#include "fflas-ffpack/utils/timer.h"  
#include "fflas-ffpack/utils/args-parser.h"
```

## Macros

- #define [\\_\\_FFLASFFPACK\\_OPENBLAS\\_NT\\_ALREADY\\_SET](#) 1

## Functions

- int [main](#) (int argc, char \*\*argv)

### 13.28.1 Macro Definition Documentation

#### 13.28.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

### 13.28.2 Function Documentation

#### 13.28.2.1 main()

```
int main (  
    int argc,  
    char ** argv)
```

## 13.29 benchmark-fsytrf.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"  
#include <iostream>  
#include <givaro/modular.h>  
#include "fflas-ffpack/fflas-ffpack.h"  
#include "fflas-ffpack/utils/timer.h"  
#include "fflas-ffpack/utils/fflas_io.h"  
#include "fflas-ffpack/utils/args-parser.h"
```

## Macros

- #define [\\_\\_FFPACK\\_FSYTRF\\_BC\\_CROUT](#)
- #define [\\_\\_FFLASFFPACK\\_OPENBLAS\\_NT\\_ALREADY\\_SET](#) 1

- `#define CUBE(x)`

## Functions

- `int main (int argc, char **argv)`

## 13.29.1 Macro Definition Documentation

### 13.29.1.1 \_\_FFPACK\_FSYTRF\_BC\_CROUT

```
#define __FFPACK_FSYTRF_BC_CROUT
```

### 13.29.1.2 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

### 13.29.1.3 CUBE

```
#define CUBE(  
            x)
```

#### Value:

```
((x)*(x)*(x))
```

## 13.29.2 Function Documentation

### 13.29.2.1 main()

```
int main (  
          int argc,  
          char ** argv)
```

## 13.30 benchmark-ftsrm-mp.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"  
#include <iostream>  
#include <vector>  
#include <string>  
#include "fflas-ffpack/utils/timer.h"  
#include "fflas-ffpack/fflas/fflas.h"  
#include "fflas-ffpack/utils/args-parser.h"  
#include "givaro/modular-integer.h"
```

## Macros

- `#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1`

## Functions

- `int main (int argc, char **argv)`

## 13.30.1 Macro Definition Documentation

### 13.30.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

## 13.30.2 Function Documentation

### 13.30.2.1 main()

```
int main (  
    int argc,  
    char ** argv)
```

## 13.31 benchmark-fftrsm.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"  
#include <iostream>  
#include <givaro/modular.h>  
#include "fflas-ffpack/fflas-ffpack.h"  
#include "fflas-ffpack/utils/timer.h"  
#include "fflas-ffpack/utils/fflas_io.h"  
#include "fflas-ffpack/utils/args-parser.h"
```

### Macros

- `#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1`

### Functions

- `int main (int argc, char **argv)`

## 13.31.1 Macro Definition Documentation

### 13.31.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

## 13.31.2 Function Documentation

### 13.31.2.1 main()

```
int main (  
    int argc,  
    char ** argv)
```

## 13.32 benchmark-fftrsv.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"  
#include <iostream>  
#include <givaro/modular.h>  
#include "fflas-ffpack/fflas-ffpack.h"  
#include "fflas-ffpack/utils/timer.h"  
#include "fflas-ffpack/utils/args-parser.h"
```

### Macros

- `#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1`

### Functions

- `int main (int argc, char **argv)`

### 13.32.1 Macro Definition Documentation

#### 13.32.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

### 13.32.2 Function Documentation

#### 13.32.2.1 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.33 benchmark-ftsrttri.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

### Macros

- `#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1`
- `#define CUBE(x)`

### Functions

- `int main (int argc, char **argv)`

### 13.33.1 Macro Definition Documentation

#### 13.33.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

#### 13.33.1.2 CUBE

```
#define CUBE(
    x)
```

### Value:

```
((x) * (x) * (x))
```

### 13.33.2 Function Documentation

#### 13.33.2.1 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.34 benchmark-inverse.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
```

```
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

## Macros

- `#define CUBE(x)`

## Functions

- `int main (int argc, char **argv)`

### 13.34.1 Macro Definition Documentation

#### 13.34.1.1 CUBE

```
#define CUBE(  
    x)
```

##### Value:

```
((x) * (x) * (x))
```

### 13.34.2 Function Documentation

#### 13.34.2.1 main()

```
int main (  
    int argc,  
    char ** argv)
```

## 13.35 benchmark-lqup-mp.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"  
#include <iostream>  
#include <vector>  
#include <string>  
#include "fflas-ffpack/utils/timer.h"  
#include "fflas-ffpack/ffpack/ffpack.h"  
#include "fflas-ffpack/utils/args-parser.h"  
#include "givaro/modular-integer.h"
```

## Functions

- `int main (int argc, char **argv)`

### 13.35.1 Function Documentation

#### 13.35.1.1 main()

```
int main (  
    int argc,  
    char ** argv)
```

## 13.36 benchmark-lqup.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

### Macros

- #define [CUBE](#)(x)

### Functions

- int [main](#) (int argc, char \*\*argv)

## 13.36.1 Macro Definition Documentation

### 13.36.1.1 CUBE

```
#define CUBE(  
            x)
```

#### Value:

```
((x) * (x) * (x))
```

## 13.36.2 Function Documentation

### 13.36.2.1 main()

```
int main (  
          int argc,  
          char ** argv)
```

## 13.37 benchmark-pluq.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <givaro/modular.h>
#include <givaro/givranditer.h>
#include <iostream>
#include "fflas-ffpack/config-blas.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/ffpack/ffpack.h"
```

### Macros

- #define [\\_\\_FFLASFFPACK\\_OPENBLAS\\_NT\\_ALREADY\\_SET](#) 1
- #define [CUBE](#)(x)

### Typedefs

- typedef [Givaro::ModularBalanced](#)< double > [Field](#)

## Functions

- void `verification_PLUQ` (const `Field` &`F`, typename `Field::Element` \*`B`, typename `Field::Element` \*`A`, `size_t` \*`P`, `size_t` \*`Q`, `size_t` `m`, `size_t` `n`, `size_t` `R`)
- void `Rec_Initialize` (`Field` &`F`, `Field::Element` \*`C`, `size_t` `m`, `size_t` `n`, `size_t` `ldc`)
- int `main` (int `argc`, char \*\*`argv`)

## 13.37.1 Macro Definition Documentation

### 13.37.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

### 13.37.1.2 CUBE

```
#define CUBE(  
            x)
```

#### Value:

```
((x)*(x)*(x))
```

## 13.37.2 Typedef Documentation

### 13.37.2.1 Field

```
typedef Givaro::ModularBalanced<double> Field
```

## 13.37.3 Function Documentation

### 13.37.3.1 verification\_PLUQ()

```
void verification_PLUQ (  
    const Field & F,  
    typename Field::Element * B,  
    typename Field::Element * A,  
    size_t * P,  
    size_t * Q,  
    size_t m,  
    size_t n,  
    size_t R)
```

### 13.37.3.2 Rec\_Initialize()

```
void Rec_Initialize (  
    Field & F,  
    Field::Element * C,  
    size_t m,  
    size_t n,  
    size_t ldc)
```

### 13.37.3.3 main()

```
int main (  
    int argc,  
    char ** argv)
```

## 13.38 benchmark-quasisep.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"  
#include <iostream>  
#include <givaro/modular.h>
```

```
#include <givaro/givpoly1.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

## Macros

- `#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1`

## Functions

- `template<class Field >`  
`void run_with_field (int q, size_t n, size_t m, size_t t, size_t r, size_t iter, uint64_t seed)`
- `int main (int argc, char **argv)`

## 13.38.1 Macro Definition Documentation

### 13.38.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

## 13.38.2 Function Documentation

### 13.38.2.1 run\_with\_field()

```
template<class Field >
void run_with_field (
    int q,
    size_t n,
    size_t m,
    size_t t,
    size_t r,
    size_t iter,
    uint64_t seed)
```

### 13.38.2.2 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.39 benchmark-storage-transpose.C File Reference

```
#include <iomanip>
#include <iostream>
#include <random>
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/fflas_memory.h"
#include <givaro/modular.h>
#include <recint/rint.h>
#include "fflas-ffpack/fflas/fflas_transpose.h"
```



**Data Structures**

- class [Bench<Elt>](#)

**Functions**

- int [main](#) (int argc, char \*\*argv)

**13.39.1 Function Documentation****13.39.1.1 main()**

```
int main (
    int argc,
    char ** argv)
```

**13.40 benchmark-wino.C File Reference**

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <fstream>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/args-parser.h"
```

**Macros**

- #define [CUBE](#)(x)

**Functions**

- template<class [Field](#) >  
void [launch\\_wino](#) (const [Field](#) &F, const size\_t &n, const size\_t &NB, const size\_t &wino, const bool &asmax, const size\_t &seed, const bool compare)
- int [main](#) (int argc, char \*\*argv)

**13.40.1 Macro Definition Documentation****13.40.1.1 CUBE**

```
#define CUBE(
    x)
```

**Value:**

```
((x) * (x) * (x))
```

**13.40.2 Function Documentation****13.40.2.1 launch\_wino()**

```
template<class Field >
void launch_wino (
    const Field & F,
    const size_t & n,
    const size_t & NB,
    const size_t & wino,
    const bool & asmax,
    const size_t & seed,
    const bool compare)
```

### 13.40.2.2 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.41 mainpage.doxy File Reference

### 13.42 det.C File Reference

```
#include <givaro/modular.h>
#include <iostream>
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/fflas_io.h"
```

#### Functions

- int [main](#) (int argc, char \*\*argv)

*This example computes the determinant of a matrix over a defined finite field.*

#### 13.42.1 Function Documentation

##### 13.42.1.1 main()

```
int main (
    int argc,
    char ** argv)
```

This example computes the determinant of a matrix over a defined finite field.  
Outputs the determinant.

## 13.43 matmul.C File Reference

```
#include <iostream>
#include "fflas-ffpack/fflas-ffpack.h"
```

#### Functions

- int [main](#) (int argc, char \*\*argv)

*This example computes the matrix multiplication over a defined finite field.*

#### 13.43.1 Function Documentation

##### 13.43.1.1 main()

```
int main (
    int argc,
    char ** argv)
```

This example computes the matrix multiplication over a defined finite field.  
Outputs the product of the matrix given as input.

## 13.44 rank.C File Reference

```
#include <iostream>
#include "fflas-ffpack/fflas-ffpack.h"
```

### Functions

- int [main](#) (int argc, char \*\*argv)

*This example computes the rank of a matrix over a defined finite field.*

### 13.44.1 Function Documentation

#### 13.44.1.1 main()

```
int main (
    int argc,
    char ** argv)
```

This example computes the rank of a matrix over a defined finite field.  
Outputs the rank.

## 13.45 solve.C File Reference

```
#include <iostream>
#include "fflas-ffpack/fflas-ffpack.h"
```

### Functions

- int [main](#) (int argc, char \*\*argv)

*This example solve the quare system defined by the input over a defined finite field.*

### 13.45.1 Function Documentation

#### 13.45.1.1 main()

```
int main (
    int argc,
    char ** argv)
```

This example solve the quare system defined by the input over a defined finite field.

## 13.46 checker\_charpoly.inl File Reference

```
#include "fflas-ffpack/ffpack/ffpack.h"
```

### Data Structures

- class [CheckerImplem\\_charpoly](#)< [Field](#), [Polynomial](#) >
- class [CheckerImplem\\_charpoly](#)< [Givaro::ZRing](#)< [Givaro::Integer](#) >, [Polynomial](#) >

### Namespaces

- namespace [FFPACK](#)

*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

## Macros

- `#define` [\\_\\_FFLASFFPACK\\_checker\\_charpoly\\_INL](#)

### 13.46.1 Macro Definition Documentation

#### 13.46.1.1 [\\_\\_FFLASFFPACK\\_checker\\_charpoly\\_INL](#)

```
#define __FFLASFFPACK_checker_charpoly_INL
```

## 13.47 checker\_det.inl File Reference

```
#include "fflas-ffpack/ffpack/ffpack.h"
```

## Data Structures

- class [CheckerImplem\\_Det< Field >](#)

## Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

## Macros

- `#define` [\\_\\_FFLASFFPACK\\_checker\\_det\\_INL](#)

### 13.47.1 Macro Definition Documentation

#### 13.47.1.1 [\\_\\_FFLASFFPACK\\_checker\\_det\\_INL](#)

```
#define __FFLASFFPACK_checker_det_INL
```

## 13.48 checker\_empty.h File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
```

## Data Structures

- struct [Checker\\_Empty< Field >](#)

## Namespaces

- namespace [FFLAS](#)

## 13.49 checker\_fgemm.inl File Reference

## Data Structures

- class [CheckerImplem\\_fgemm< Field >](#)

## Namespaces

- namespace [FFLAS](#)

**Macros**

- #define [\\_\\_FFLASFFPACK\\_checker\\_fgemm\\_INL](#)

**13.49.1 Macro Definition Documentation****13.49.1.1 \_\_FFLASFFPACK\_checker\_fgemm\_INL**

```
#define __FFLASFFPACK_checker_fgemm_INL
```

**13.50 checker\_ftsm.inl File Reference****Data Structures**

- class [CheckerImplem\\_ftsm](#)< Field >

**Namespaces**

- namespace [FFLAS](#)

**Macros**

- #define [\\_\\_FFLASFFPACK\\_checker\\_ftsm\\_INL](#)

**13.50.1 Macro Definition Documentation****13.50.1.1 \_\_FFLASFFPACK\_checker\_ftsm\_INL**

```
#define __FFLASFFPACK_checker_ftsm_INL
```

**13.51 checker\_invert.inl File Reference****Data Structures**

- class [CheckerImplem\\_invert](#)< Field >

**Namespaces**

- namespace [FFPACK](#)

*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

**Macros**

- #define [\\_\\_FFLASFFPACK\\_checker\\_invert\\_INL](#)

**13.51.1 Macro Definition Documentation****13.51.1.1 \_\_FFLASFFPACK\_checker\_invert\_INL**

```
#define __FFLASFFPACK_checker_invert_INL
```

**13.52 checker\_pluq.inl File Reference**

```
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/fflas_io.h"
```

## Data Structures

- class [CheckerImplem\\_PLUQ<Field>](#)

## Namespaces

- namespace [FFPACK](#)  
*Finite **Field** **PACK** Set of elimination based routines for dense linear algebra.*

## Macros

- `#define` [\\_\\_FFLASFFPACK\\_checker\\_pluq\\_INL](#)

## 13.52.1 Macro Definition Documentation

### 13.52.1.1 [\\_\\_FFLASFFPACK\\_checker\\_pluq\\_INL](#)

```
#define __FFLASFFPACK_checker_pluq_INL
```

## 13.53 checkers.doxy File Reference

## 13.54 checkers\_fflas.h File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "checker_empty.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/fflas/fflas_enum.h"
#include "fflas-ffpack/fflas/fflas_memory.h"
```

## Data Structures

- class [FailureFgemmCheck](#)
- class [FailureTrsmCheck](#)

## Namespaces

- namespace [FFLAS](#)

## Typedefs

- `template<class Field>`  
  `using Checker\_fgemm = FFLAS::Checker\_Empty<Field>`
- `template<class Field>`  
  `using Checker\_ftrsm = FFLAS::Checker\_Empty<Field>`

## 13.55 checkers\_fflas.inl File Reference

```
#include "checker_fgemm.inl"
#include "checker_ftrsm.inl"
```

## Namespaces

- namespace [FFLAS](#)

## Macros

- `#define FFLASFFPACK_checkers_fflas_inl_H`

## Typedefs

- `template<class Field >`  
`using ForceCheck_fgemm = CheckerImplem_fgemm<Field>`
- `template<class Field >`  
`using ForceCheck_ftrsm = CheckerImplem_ftrsm<Field>`

## 13.55.1 Macro Definition Documentation

### 13.55.1.1 FFLASFFPACK\_checkers\_fflas\_inl\_H

```
#define FFLASFFPACK_checkers_fflas_inl_H
```

## 13.56 checkers\_ffpack.h File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "checker_empty.h"
#include "fflas-ffpack/ffpack/ffpack.h"
```

## Data Structures

- class [FailurePLUQCheck](#)
- class [FailureDetCheck](#)
- class [FailureInvertCheck](#)
- class [FailureCharpolyCheck](#)

## Namespaces

- namespace [FFPACK](#)

*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

## Typedefs

- `template<class Field >`  
`using Checker_PLUQ = FFLAS::Checker_Empty<Field>`
- `template<class Field >`  
`using Checker_Det = FFLAS::Checker_Empty<Field>`
- `template<class Field >`  
`using Checker_invert = FFLAS::Checker_Empty<Field>`
- `template<class Field , class Polynomial >`  
`using Checker_charpoly = FFLAS::Checker_Empty<Field>`

## 13.57 checkers\_ffpack.inl File Reference

```
#include "checker_pluq.inl"
#include "checker_det.inl"
#include "checker_invert.inl"
#include "checker_charpoly.inl"
```

## Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

## Macros

- `#define` [FFLASFFPACK\\_checkers\\_ffpack\\_inl\\_H](#)

## Typedefs

- `template<class Field >`  
  `using ForceCheck\_PLUQ = CheckerImplem\_PLUQ<Field>`
- `template<class Field >`  
  `using ForceCheck\_Det = CheckerImplem\_Det<Field>`
- `template<class Field >`  
  `using ForceCheck\_invert = CheckerImplem\_invert<Field>`
- `template<class Field , class Polynomial >`  
  `using ForceCheck\_charpoly = CheckerImplem\_charpoly<Field,Polynomial>`

## 13.57.1 Macro Definition Documentation

### 13.57.1.1 FFLASFFPACK\_checkers\_ffpack\_inl\_H

```
#define FFLASFFPACK_checkers_ffpack_inl_H
```

## 13.58 config-blas.h File Reference

### Macros

- `#define` [CBLAS\\_INT](#) int
- `#define` [CBLAS\\_ENUM\\_DEFINED\\_H](#)
- `#define` [CBLAS\\_EXTERNALS](#)
- `#define` [blas\\_enum](#) enum

### Enumerations

- enum [CBLAS\\_ORDER](#) { [CblasRowMajor](#) =101 , [CblasColMajor](#) =102 }
- enum [CBLAS\\_TRANSPOSE](#) { [CblasNoTrans](#) =111 , [CblasTrans](#) =112 , [CblasConjTrans](#) =113 , [AtlasConj](#) =114 }
- enum [CBLAS\\_UPLO](#) { [CblasUpper](#) =121 , [CblasLower](#) =122 }
- enum [CBLAS\\_DIAG](#) { [CblasNonUnit](#) =131 , [CblasUnit](#) =132 }
- enum [CBLAS\\_SIDE](#) { [CblasLeft](#) =141 , [CblasRight](#) =142 }

### Functions

- void [daxpy\\_](#) (const int \*, const double \*, const double \*, const int \*, double \*, const int \*)
- void [saxpy\\_](#) (const int \*, const float \*, const float \*, const int \*, float \*, const int \*)
- double [ddot\\_](#) (const int \*, const double \*, const int \*, const double \*, const int \*)
- float [sdot\\_](#) (const int \*, const float \*, const int \*, const float \*, const int \*)
- double [dasum\\_](#) (const int \*, const double \*, const int \*)
- int [idamax\\_](#) (const int \*, const double \*, const int \*)
- double [dnrm2\\_](#) (const int \*, const double \*, const int \*)
- void [dgemv\\_](#) (const char \*, const int \*, const int \*, const double \*, const double \*, const int \*, const double \*, const int \*, const double \*, double \*, const int \*)
- void [sgemv\\_](#) (const char \*, const int \*, const int \*, const float \*, const float \*, const int \*, const float \*, const int \*, const float \*, float \*, const int \*)



- void [dger\\_](#) (const int \*, const int \*, const double \*, const double \*, const int \*, const double \*, const int \*, double \*, const int \*)
- void [sger\\_](#) (const int \*, const int \*, const float \*, const float \*, const int \*, const float \*, const int \*, float \*, const int \*)
- void [dcopy\\_](#) (const int \*, const double \*, const int \*, double \*, const int \*)
- void [scopy\\_](#) (const int \*, const float \*, const int \*, float \*, const int \*)
- void [dscal\\_](#) (const int \*, const double \*, double \*, const int \*)
- void [sscal\\_](#) (const int \*, const float \*, float \*, const int \*)
- void [dtrsm\\_](#) (const char \*, const char \*, const char \*, const char \*, const int \*, const int \*, const double \*, const double \*, const int \*, double \*, const int \*)
- void [strsm\\_](#) (const char \*, const char \*, const char \*, const char \*, const int \*, const int \*, const float \*, const float \*, const int \*, float \*, const int \*)
- void [dtrmm\\_](#) (const char \*, const char \*, const char \*, const char \*, const int \*, const int \*, const double \*, const double \*, const int \*, double \*, const int \*)
- void [strmm\\_](#) (const char \*, const char \*, const char \*, const char \*, const int \*, const int \*, const float \*, const float \*, const int \*, float \*, const int \*)
- void [sgemm\\_](#) (const char \*, const char \*, const int \*, const int \*, const int \*, const float \*, const float \*, const int \*, const float \*, const int \*, const float \*, float \*, const int \*)
- void [dgemm\\_](#) (const char \*, const char \*, const int \*, const int \*, const int \*, const double \*, const double \*, const int \*, const double \*, const int \*, const double \*, double \*, const int \*)
- void [cblas\\_dsyrk](#) (const enum [CBLAS\\_ORDER](#) Order, const enum [CBLAS\\_UPLO](#) Uplo, const enum [CBLAS\\_TRANSPOSE](#) Trans, const int N, const int K, const double alpha, const double \*A, const int lda, const double beta, double \*C, const int ldc)

## 13.58.1 Macro Definition Documentation

### 13.58.1.1 CBLAS\_INT

```
#define CBLAS_INT int
```

### 13.58.1.2 CBLAS\_ENUM\_DEFINED\_H

```
#define CBLAS_ENUM_DEFINED_H
```

### 13.58.1.3 CBLAS\_EXTERNALS

```
#define CBLAS_EXTERNALS
```

### 13.58.1.4 blas\_enum

```
#define blas_enum enum
```

## 13.58.2 Enumeration Type Documentation

### 13.58.2.1 CBLAS\_ORDER

```
enum CBLAS\_ORDER
```

Enumerator

|               |  |
|---------------|--|
| CblasRowMajor |  |
| CblasColMajor |  |

### 13.58.2.2 CBLAS\_TRANSPOSE

```
enum CBLAS\_TRANSPOSE
```

**Enumerator**

|                |  |
|----------------|--|
| CblasNoTrans   |  |
| CblasTrans     |  |
| CblasConjTrans |  |
| AtlasConj      |  |

**13.58.2.3 CBLAS\_UPLO**

enum CBLAS\_UPLO

**Enumerator**

|            |  |
|------------|--|
| CblasUpper |  |
| CblasLower |  |

**13.58.2.4 CBLAS\_DIAG**

enum CBLAS\_DIAG

**Enumerator**

|              |  |
|--------------|--|
| CblasNonUnit |  |
| CblasUnit    |  |

**13.58.2.5 CBLAS\_SIDE**

enum CBLAS\_SIDE

**Enumerator**

|            |  |
|------------|--|
| CblasLeft  |  |
| CblasRight |  |

**13.58.3 Function Documentation****13.58.3.1 daxpy\_()**

```
void daxpy_ (
    const int * ,
    const double * ,
    const double * ,
    const int * ,
    double * ,
    const int * )
```

**13.58.3.2 saxpy\_()**

```
void saxpy_ (
    const int * ,
    const float * ,
    const float * ,
    const int * ,
    float * ,
    const int * )
```

### 13.58.3.3 ddot\_()

```
double ddot_ (
    const int * ,
    const double * ,
    const int * ,
    const double * ,
    const int * )
```

### 13.58.3.4 sdot\_()

```
float sdot_ (
    const int * ,
    const float * ,
    const int * ,
    const float * ,
    const int * )
```

### 13.58.3.5 dasum\_()

```
double dasum_ (
    const int * ,
    const double * ,
    const int * )
```

### 13.58.3.6 idamax\_()

```
int idamax_ (
    const int * ,
    const double * ,
    const int * )
```

### 13.58.3.7 dnorm2\_()

```
double dnorm2_ (
    const int * ,
    const double * ,
    const int * )
```

### 13.58.3.8 dgemv\_()

```
void dgemv_ (
    const char * ,
    const int * ,
    const int * ,
    const double * ,
    const double * ,
    const int * ,
    const double * ,
    const int * ,
    const double * ,
    double * ,
    const int * )
```

### 13.58.3.9 sgemv\_()

```
void sgemv_ (
    const char * ,
    const int * ,
    const int * ,
```

```
    const float * ,  
    const float * ,  
    const int * ,  
    const float * ,  
    const int * ,  
    const float * ,  
    float * ,  
    const int * )
```

#### 13.58.3.10 dger\_()

```
void dger_ (  
    const int * ,  
    const int * ,  
    const double * ,  
    const double * ,  
    const int * ,  
    const double * ,  
    const int * ,  
    double * ,  
    const int * )
```

#### 13.58.3.11 sger\_()

```
void sger_ (  
    const int * ,  
    const int * ,  
    const float * ,  
    const float * ,  
    const int * ,  
    const float * ,  
    const int * ,  
    float * ,  
    const int * )
```

#### 13.58.3.12 dcopy\_()

```
void dcopy_ (  
    const int * ,  
    const double * ,  
    const int * ,  
    double * ,  
    const int * )
```

#### 13.58.3.13 scopy\_()

```
void scopy_ (  
    const int * ,  
    const float * ,  
    const int * ,  
    float * ,  
    const int * )
```

#### 13.58.3.14 dscal\_()

```
void dscal_ (  
    const int * ,  
    const double * ,
```

```
double * ,  
const int * )
```

#### 13.58.3.15 sscal\_()

```
void sscal_ (  
    const int * ,  
    const float * ,  
    float * ,  
    const int * )
```

#### 13.58.3.16 dtrsm\_()

```
void dtrsm_ (  
    const char * ,  
    const char * ,  
    const char * ,  
    const char * ,  
    const int * ,  
    const int * ,  
    const double * ,  
    const double * ,  
    const int * ,  
    double * ,  
    const int * )
```

#### 13.58.3.17 strsm\_()

```
void strsm_ (  
    const char * ,  
    const char * ,  
    const char * ,  
    const char * ,  
    const int * ,  
    const int * ,  
    const float * ,  
    const float * ,  
    const int * ,  
    float * ,  
    const int * )
```

#### 13.58.3.18 dtrmm\_()

```
void dtrmm_ (  
    const char * ,  
    const char * ,  
    const char * ,  
    const char * ,  
    const int * ,  
    const int * ,  
    const double * ,  
    const double * ,  
    const int * ,  
    double * ,  
    const int * )
```

#### 13.58.3.19 strmm\_()

```
void strmm_ (  

```

```
    const char * ,  
    const char * ,  
    const char * ,  
    const char * ,  
    const int * ,  
    const int * ,  
    const float * ,  
    const float * ,  
    const int * ,  
    float * ,  
    const int * )
```

#### 13.58.3.20 sgemm\_()

```
void sgemm_ (  
    const char * ,  
    const char * ,  
    const int * ,  
    const int * ,  
    const int * ,  
    const float * ,  
    const float * ,  
    const int * ,  
    const float * ,  
    const int * ,  
    const float * ,  
    float * ,  
    const int * )
```

#### 13.58.3.21 dgemm\_()

```
void dgemm_ (  
    const char * ,  
    const char * ,  
    const int * ,  
    const int * ,  
    const int * ,  
    const double * ,  
    const double * ,  
    const int * ,  
    const double * ,  
    const int * ,  
    const double * ,  
    double * ,  
    const int * )
```

#### 13.58.3.22 cblas\_dsyrk()

```
void cblas_dsyrk (  
    const enum CBLAS_ORDER Order,  
    const enum CBLAS_UPLO Uplo,  
    const enum CBLAS_TRANSPOSE Trans,  
    const int N,  
    const int K,  
    const double alpha,  
    const double * A,  
    const int lda,  
    const double beta,
```

```
double * C,
const int ldc)
```

## 13.59 config.h File Reference

### Macros

- #define [HAVE\\_BLAS](#) 1
- #define [HAVE\\_CBLAS](#) 1
- #define [HAVE\\_CXX11](#) 1
- #define [HAVE\\_DLFCN\\_H](#) 1
- #define [HAVE\\_FLOAT\\_H](#) 1
- #define [HAVE\\_INT128](#) 1
- #define [HAVE\\_INTPTR\\_T](#) 1
- #define [HAVE\\_INTTYPES\\_H](#) 1
- #define [HAVE\\_LAPACK](#) 1
- #define [HAVE\\_LIMITS\\_H](#) 1
- #define [HAVE\\_LITTLE\\_ENDIAN](#) 1
- #define [HAVE\\_PTHREAD\\_H](#) 1
- #define [HAVE\\_STDDEF\\_H](#) 1
- #define [HAVE\\_STDINT\\_H](#) 1
- #define [HAVE\\_STDIO\\_H](#) 1
- #define [HAVE\\_STDLIB\\_H](#) 1
- #define [HAVE\\_STRINGS\\_H](#) 1
- #define [HAVE\\_STRING\\_H](#) 1
- #define [HAVE\\_SYS\\_STAT\\_H](#) 1
- #define [HAVE\\_SYS\\_TIME\\_H](#) 1
- #define [HAVE\\_SYS\\_TYPES\\_H](#) 1
- #define [HAVE\\_UNISTD\\_H](#) 1
- #define [LT\\_OBJDIR](#) ".libs/"
- #define [OPENBLAS\\_NUM\\_THREADS](#) 1
- #define [PACKAGE](#) "fflas-ffpack"
- #define [PACKAGE\\_BUGREPORT](#) "ffpack-devel@googlegroups.com"
- #define [PACKAGE\\_NAME](#) "FFLAS-FFPACK"
- #define [PACKAGE\\_STRING](#) "FFLAS-FFPACK 2.5.0"
- #define [PACKAGE\\_TARNAME](#) "fflas-ffpack"
- #define [PACKAGE\\_URL](#) "https://github.com/linbox-team/fflas-ffpack"
- #define [PACKAGE\\_VERSION](#) "2.5.0"
- #define [SIZEOF\\_CHAR](#) 1
- #define [SIZEOF\\_INT](#) 4
- #define [SIZEOF\\_LONG](#) 8
- #define [SIZEOF\\_LONG\\_LONG](#) 8
- #define [SIZEOF\\_SHORT](#) 2
- #define [SIZEOF\\_\\_INT64\\_T](#) 8
- #define [STDC\\_HEADERS](#) 1
- #define [USE\\_OPENMP](#) 1
- #define [VERSION](#) "2.5.0"

### 13.59.1 Macro Definition Documentation

#### 13.59.1.1 HAVE\_BLAS

```
#define HAVE_BLAS 1
```

#### 13.59.1.2 HAVE\_CBLAS

```
#define HAVE_CBLAS 1
```

**13.59.1.3 HAVE\_CXX11**

```
#define HAVE_CXX11 1
```

**13.59.1.4 HAVE\_DLFCN\_H**

```
#define HAVE_DLFCN_H 1
```

**13.59.1.5 HAVE\_FLOAT\_H**

```
#define HAVE_FLOAT_H 1
```

**13.59.1.6 HAVE\_INT128**

```
#define HAVE_INT128 1
```

**13.59.1.7 HAVE\_INTPYPES\_H**

```
#define HAVE_INTPYPES_H 1
```

**13.59.1.8 HAVE\_LAPACK**

```
#define HAVE_LAPACK 1
```

**13.59.1.9 HAVE\_LIMITS\_H**

```
#define HAVE_LIMITS_H 1
```

**13.59.1.10 HAVE\_LITTLE\_ENDIAN**

```
#define HAVE_LITTLE_ENDIAN 1
```

**13.59.1.11 HAVE\_PTHREAD\_H**

```
#define HAVE_PTHREAD_H 1
```

**13.59.1.12 HAVE\_STDDEF\_H**

```
#define HAVE_STDDEF_H 1
```

**13.59.1.13 HAVE\_STDINT\_H**

```
#define HAVE_STDINT_H 1
```

**13.59.1.14 HAVE\_STDIO\_H**

```
#define HAVE_STDIO_H 1
```

**13.59.1.15 HAVE\_STDLIB\_H**

```
#define HAVE_STDLIB_H 1
```

**13.59.1.16 HAVE\_STRINGS\_H**

```
#define HAVE_STRINGS_H 1
```

**13.59.1.17 HAVE\_STRING\_H**

```
#define HAVE_STRING_H 1
```



**13.59.1.18 HAVE\_SYS\_STAT\_H**

```
#define HAVE_SYS_STAT_H 1
```

**13.59.1.19 HAVE\_SYS\_TIME\_H**

```
#define HAVE_SYS_TIME_H 1
```

**13.59.1.20 HAVE\_SYS\_TYPES\_H**

```
#define HAVE_SYS_TYPES_H 1
```

**13.59.1.21 HAVE\_UNISTD\_H**

```
#define HAVE_UNISTD_H 1
```

**13.59.1.22 LT\_OBJDIR**

```
#define LT_OBJDIR ".libs/"
```

**13.59.1.23 OPENBLAS\_NUM\_THREADS**

```
#define OPENBLAS_NUM_THREADS 1
```

**13.59.1.24 PACKAGE**

```
#define PACKAGE "fflas-ffpack"
```

**13.59.1.25 PACKAGE\_BUGREPORT**

```
#define PACKAGE_BUGREPORT "ffpack-devel@googlegroups.com"
```

**13.59.1.26 PACKAGE\_NAME**

```
#define PACKAGE_NAME "FFLAS-FFPACK"
```

**13.59.1.27 PACKAGE\_STRING**

```
#define PACKAGE_STRING "FFLAS-FFPACK 2.5.0"
```

**13.59.1.28 PACKAGE\_TARNAME**

```
#define PACKAGE_TARNAME "fflas-ffpack"
```

**13.59.1.29 PACKAGE\_URL**

```
#define PACKAGE_URL "https://github.com/linbox-team/fflas-ffpack"
```

**13.59.1.30 PACKAGE\_VERSION**

```
#define PACKAGE_VERSION "2.5.0"
```

**13.59.1.31 SIZEOF\_CHAR**

```
#define SIZEOF_CHAR 1
```

**13.59.1.32 SIZEOF\_INT**

```
#define SIZEOF_INT 4
```

**13.59.1.33    `SIZEOF_LONG`**

```
#define SIZEOF_LONG 8
```

**13.59.1.34    `SIZEOF_LONG_LONG`**

```
#define SIZEOF_LONG_LONG 8
```

**13.59.1.35    `SIZEOF_SHORT`**

```
#define SIZEOF_SHORT 2
```

**13.59.1.36    `SIZEOF___INT64_T`**

```
#define SIZEOF___INT64_T 8
```

**13.59.1.37    `STDC_HEADERS`**

```
#define STDC_HEADERS 1
```

**13.59.1.38    `USE_OPENMP`**

```
#define USE_OPENMP 1
```

**13.59.1.39    `VERSION`**

```
#define VERSION "2.5.0"
```

**13.60    `fflas-ffpack/config.h` File Reference****Macros**

- `#define __FFLASFFPACK_HAVE_BLAS 1`
- `#define __FFLASFFPACK_HAVE_CBLAS 1`
- `#define __FFLASFFPACK_HAVE_CXX11 1`
- `#define __FFLASFFPACK_HAVE_DLFCN_H 1`
- `#define __FFLASFFPACK_HAVE_FLOAT_H 1`
- `#define __FFLASFFPACK_HAVE_INT128 1`
- `#define __FFLASFFPACK_HAVE_INTTYPES_H 1`
- `#define __FFLASFFPACK_HAVE_LAPACK 1`
- `#define __FFLASFFPACK_HAVE_LIMITS_H 1`
- `#define __FFLASFFPACK_HAVE_LITTLE_ENDIAN 1`
- `#define __FFLASFFPACK_HAVE_PTHREAD_H 1`
- `#define __FFLASFFPACK_HAVE_STDDEF_H 1`
- `#define __FFLASFFPACK_HAVE_STDINT_H 1`
- `#define __FFLASFFPACK_HAVE_STDIO_H 1`
- `#define __FFLASFFPACK_HAVE_STDLIB_H 1`
- `#define __FFLASFFPACK_HAVE_STRINGS_H 1`
- `#define __FFLASFFPACK_HAVE_STRING_H 1`
- `#define __FFLASFFPACK_HAVE_SYS_STAT_H 1`
- `#define __FFLASFFPACK_HAVE_SYS_TIME_H 1`
- `#define __FFLASFFPACK_HAVE_SYS_TYPES_H 1`
- `#define __FFLASFFPACK_HAVE_UNISTD_H 1`
- `#define __FFLASFFPACK_LT_OBJDIR ".libs/"`
- `#define __FFLASFFPACK_OPENBLAS_NUM_THREADS 1`
- `#define __FFLASFFPACK_PACKAGE "fflas-ffpack"`
- `#define __FFLASFFPACK_PACKAGE_BUGREPORT "ffpack-devel@googlegroups.com"`

- `#define __FFLASFFPACK_PACKAGE_NAME "FFLAS-FFPACK"`
- `#define __FFLASFFPACK_PACKAGE_STRING "FFLAS-FFPACK 2.5.0"`
- `#define __FFLASFFPACK_PACKAGE_TARNAME "fflas-ffpack"`
- `#define __FFLASFFPACK_PACKAGE_URL "https://github.com/linbox-team/fflas-ffpack"`
- `#define __FFLASFFPACK_PACKAGE_VERSION "2.5.0"`
- `#define __FFLASFFPACK_SIZEOF_CHAR 1`
- `#define __FFLASFFPACK_SIZEOF_INT 4`
- `#define __FFLASFFPACK_SIZEOF_LONG 8`
- `#define __FFLASFFPACK_SIZEOF_LONG_LONG 8`
- `#define __FFLASFFPACK_SIZEOF_SHORT 2`
- `#define __FFLASFFPACK_SIZEOF__INT64_T 8`
- `#define __FFLASFFPACK_STDC_HEADERS 1`
- `#define __FFLASFFPACK_USE_OPENMP 1`
- `#define __FFLASFFPACK_VERSION "2.5.0"`

## 13.60.1 Macro Definition Documentation

### 13.60.1.1 \_\_FFLASFFPACK\_HAVE\_BLAS

```
#define __FFLASFFPACK_HAVE_BLAS 1
```

### 13.60.1.2 \_\_FFLASFFPACK\_HAVE\_CBLAS

```
#define __FFLASFFPACK_HAVE_CBLAS 1
```

### 13.60.1.3 \_\_FFLASFFPACK\_HAVE\_CXX11

```
#define __FFLASFFPACK_HAVE_CXX11 1
```

### 13.60.1.4 \_\_FFLASFFPACK\_HAVE\_DLFCN\_H

```
#define __FFLASFFPACK_HAVE_DLFCN_H 1
```

### 13.60.1.5 \_\_FFLASFFPACK\_HAVE\_FLOAT\_H

```
#define __FFLASFFPACK_HAVE_FLOAT_H 1
```

### 13.60.1.6 \_\_FFLASFFPACK\_HAVE\_INT128

```
#define __FFLASFFPACK_HAVE_INT128 1
```

### 13.60.1.7 \_\_FFLASFFPACK\_HAVE\_INTPYPES\_H

```
#define __FFLASFFPACK_HAVE_INTPYPES_H 1
```

### 13.60.1.8 \_\_FFLASFFPACK\_HAVE\_LAPACK

```
#define __FFLASFFPACK_HAVE_LAPACK 1
```

### 13.60.1.9 \_\_FFLASFFPACK\_HAVE\_LIMITS\_H

```
#define __FFLASFFPACK_HAVE_LIMITS_H 1
```

### 13.60.1.10 \_\_FFLASFFPACK\_HAVE\_LITTLE\_ENDIAN

```
#define __FFLASFFPACK_HAVE_LITTLE_ENDIAN 1
```

### 13.60.1.11 \_\_FFLASFFPACK\_HAVE\_PTHREAD\_H

```
#define __FFLASFFPACK_HAVE_PTHREAD_H 1
```

**13.60.1.12 \_\_FFLASFFPACK\_HAVE\_STDDEF\_H**

```
#define __FFLASFFPACK_HAVE_STDDEF_H 1
```

**13.60.1.13 \_\_FFLASFFPACK\_HAVE\_STDINT\_H**

```
#define __FFLASFFPACK_HAVE_STDINT_H 1
```

**13.60.1.14 \_\_FFLASFFPACK\_HAVE\_STDIO\_H**

```
#define __FFLASFFPACK_HAVE_STDIO_H 1
```

**13.60.1.15 \_\_FFLASFFPACK\_HAVE\_STDLIB\_H**

```
#define __FFLASFFPACK_HAVE_STDLIB_H 1
```

**13.60.1.16 \_\_FFLASFFPACK\_HAVE\_STRINGS\_H**

```
#define __FFLASFFPACK_HAVE_STRINGS_H 1
```

**13.60.1.17 \_\_FFLASFFPACK\_HAVE\_STRING\_H**

```
#define __FFLASFFPACK_HAVE_STRING_H 1
```

**13.60.1.18 \_\_FFLASFFPACK\_HAVE\_SYS\_STAT\_H**

```
#define __FFLASFFPACK_HAVE_SYS_STAT_H 1
```

**13.60.1.19 \_\_FFLASFFPACK\_HAVE\_SYS\_TIME\_H**

```
#define __FFLASFFPACK_HAVE_SYS_TIME_H 1
```

**13.60.1.20 \_\_FFLASFFPACK\_HAVE\_SYS\_TYPES\_H**

```
#define __FFLASFFPACK_HAVE_SYS_TYPES_H 1
```

**13.60.1.21 \_\_FFLASFFPACK\_HAVE\_UNISTD\_H**

```
#define __FFLASFFPACK_HAVE_UNISTD_H 1
```

**13.60.1.22 \_\_FFLASFFPACK\_LT\_OBJDIR**

```
#define __FFLASFFPACK_LT_OBJDIR ".libs/"
```

**13.60.1.23 \_\_FFLASFFPACK\_OPENBLAS\_NUM\_THREADS**

```
#define __FFLASFFPACK_OPENBLAS_NUM_THREADS 1
```

**13.60.1.24 \_\_FFLASFFPACK\_PACKAGE**

```
#define __FFLASFFPACK_PACKAGE "fflas-ffpack"
```

**13.60.1.25 \_\_FFLASFFPACK\_PACKAGE\_BUGREPORT**

```
#define __FFLASFFPACK_PACKAGE_BUGREPORT "ffpack-devel@googlegroups.com"
```

**13.60.1.26 \_\_FFLASFFPACK\_PACKAGE\_NAME**

```
#define __FFLASFFPACK_PACKAGE_NAME "FFLAS-FFPACK"
```

**13.60.1.27 \_\_FFLASFFPACK\_PACKAGE\_STRING**

```
#define __FFLASFFPACK_PACKAGE_STRING "FFLAS-FFPACK 2.5.0"
```

**13.60.1.28 \_\_FFLASFFPACK\_PACKAGE\_TARNAME**

```
#define __FFLASFFPACK_PACKAGE_TARNAME "fflas-ffpack"
```

**13.60.1.29 \_\_FFLASFFPACK\_PACKAGE\_URL**

```
#define __FFLASFFPACK_PACKAGE_URL "https://github.com/linbox-team/fflas-ffpack"
```

**13.60.1.30 \_\_FFLASFFPACK\_PACKAGE\_VERSION**

```
#define __FFLASFFPACK_PACKAGE_VERSION "2.5.0"
```

**13.60.1.31 \_\_FFLASFFPACK\_SIZEOF\_CHAR**

```
#define __FFLASFFPACK_SIZEOF_CHAR 1
```

**13.60.1.32 \_\_FFLASFFPACK\_SIZEOF\_INT**

```
#define __FFLASFFPACK_SIZEOF_INT 4
```

**13.60.1.33 \_\_FFLASFFPACK\_SIZEOF\_LONG**

```
#define __FFLASFFPACK_SIZEOF_LONG 8
```

**13.60.1.34 \_\_FFLASFFPACK\_SIZEOF\_LONG\_LONG**

```
#define __FFLASFFPACK_SIZEOF_LONG_LONG 8
```

**13.60.1.35 \_\_FFLASFFPACK\_SIZEOF\_SHORT**

```
#define __FFLASFFPACK_SIZEOF_SHORT 2
```

**13.60.1.36 \_\_FFLASFFPACK\_SIZEOF\_\_INT64\_T**

```
#define __FFLASFFPACK_SIZEOF__INT64_T 8
```

**13.60.1.37 \_\_FFLASFFPACK\_STDC\_HEADERS**

```
#define __FFLASFFPACK_STDC_HEADERS 1
```

**13.60.1.38 \_\_FFLASFFPACK\_USE\_OPENMP**

```
#define __FFLASFFPACK_USE_OPENMP 1
```

**13.60.1.39 \_\_FFLASFFPACK\_VERSION**

```
#define __FFLASFFPACK_VERSION "2.5.0"
```

**13.61 fflas-ffpack-config.h File Reference**

Defaults for optimised values.

```
#include "fflas-ffpack/config.h"
#include "fflas-ffpack/fflas-ffpack-thresholds.h"
#include "fflas-ffpack/fflas-ffpack-default-thresholds.h"
#include "givaro/givconfig.h"
```

**Macros**

- `#define GCC_VERSION (__GNUC__ * 10000 + __GNUC_MINOR__ * 100 + __GNUC_PATCHLEVEL__)`

**13.61.1 Detailed Description**

Defaults for optimised values.

While `fflas-ffpack-optimise.h` is created by `configure` script, (either left blank or filled by optimiser), this file produces the defaults for the optimised values. If `fflas-ffpack-optimise.h` is not empty, then its values preceeds the defaults here.

**13.61.2 Macro Definition Documentation****13.61.2.1 GCC\_VERSION**

```
#define GCC_VERSION (__GNUC__ * 10000 + __GNUC_MINOR__ * 100 + __GNUC_PATCHLEVEL__)
```

**13.62 fflas-ffpack-default-thresholds.h File Reference****Macros**

- `#define __FFLASFFPACK_WINOTHRESHOLD 1000`
- `#define __FFLASFFPACK_WINOTHRESHOLD_FLT 2000`
- `#define __FFLASFFPACK_WINOTHRESHOLD_BAL 1000`
- `#define __FFLASFFPACK_WINOTHRESHOLD_BAL_FLT 2000`
- `#define __FFLASFFPACK_PLUQ_THRESHOLD 256`
- `#define __FFLASFFPACK_CHARPOLY_LUKrylov_ArithProg_THRESHOLD 1000`
- `#define __FFLASFFPACK_CHARPOLY_Danilevskii_LUKrylov_THRESHOLD 16`
- `#define __FFLASFFPACK_ARITHPROG_THRESHOLD 30`
- `#define __FFLASFFPACK_FTRTRI_THRESHOLD 32`
- `#define __FFLASFFPACK_FSYTRF_THRESHOLD 64`
- `#define __FFLASFFPACK_FSYRK_THRESHOLD 3000`

**13.62.1 Macro Definition Documentation****13.62.1.1 \_\_FFLASFFPACK\_WINOTHRESHOLD**

```
#define __FFLASFFPACK_WINOTHRESHOLD 1000
```

**13.62.1.2 \_\_FFLASFFPACK\_WINOTHRESHOLD\_FLT**

```
#define __FFLASFFPACK_WINOTHRESHOLD_FLT 2000
```

**13.62.1.3 \_\_FFLASFFPACK\_WINOTHRESHOLD\_BAL**

```
#define __FFLASFFPACK_WINOTHRESHOLD_BAL 1000
```

**13.62.1.4 \_\_FFLASFFPACK\_WINOTHRESHOLD\_BAL\_FLT**

```
#define __FFLASFFPACK_WINOTHRESHOLD_BAL_FLT 2000
```

**13.62.1.5 \_\_FFLASFFPACK\_PLUQ\_THRESHOLD**

```
#define __FFLASFFPACK_PLUQ_THRESHOLD 256
```

**13.62.1.6 \_\_FFLASFFPACK\_CHARPOLY\_LUKrylov\_ArithProg\_THRESHOLD**

```
#define __FFLASFFPACK_CHARPOLY_LUKrylov_ArithProg_THRESHOLD 1000
```

**13.62.1.7 \_\_FFLASFFPACK\_CHARPOLY\_Danilevskii\_LUKrylov\_THRESHOLD**

```
#define __FFLASFFPACK_CHARPOLY_Danilevskii_LUKrylov_THRESHOLD 16
```

**13.62.1.8 \_\_FFLASFFPACK\_ARITHPROG\_THRESHOLD**

```
#define __FFLASFFPACK_ARITHPROG_THRESHOLD 30
```

**13.62.1.9 \_\_FFLASFFPACK\_FTRTRI\_THRESHOLD**

```
#define __FFLASFFPACK_FTRTRI_THRESHOLD 32
```

**13.62.1.10 \_\_FFLASFFPACK\_FSYTRF\_THRESHOLD**

```
#define __FFLASFFPACK_FSYTRF_THRESHOLD 64
```

**13.62.1.11 \_\_FFLASFFPACK\_FSYRK\_THRESHOLD**

```
#define __FFLASFFPACK_FSYRK_THRESHOLD 3000
```

**13.63 fflas-ffpack-thresholds.h File Reference****13.64 fflas-ffpack.doxy File Reference****13.65 fflas-ffpack.h File Reference**

Includes [FFLAS](#) and [FFPACK](#).

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas/fflas.h"
#include "ffpack/ffpack.h"
```

**13.65.1 Detailed Description**

Includes [FFLAS](#) and [FFPACK](#).

**13.66 fflas.doxy File Reference****13.67 fflas.h File Reference**

**Finite Field Linear Algebra Subroutines**

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/config.h"
#include "fflas-ffpack/config-blas.h"
#include <cmath>
#include <cstring>
#include <float.h>
#include <algorithm>
#include "fflas_enum.h"
#include "fflas-ffpack/utils/fflas_memory.h"
#include "fflas-ffpack/paladin/parallel.h"
#include "fflas_level1.inl"
#include "fflas_level2.inl"
#include "fflas_level3.inl"
#include "fflas-ffpack/checkers/checkers_fflas.h"
#include "fflas_freduce.h"
#include "fflas_fadd.h"
```

```
#include "fflas_fscal.h"
#include "fflas_fassign.h"
#include "fflas_fgemm.inl"
#include "fflas_pfgemm.inl"
#include "fflas_fgemv.inl"
#include "fflas-ffpack/paladin/pfgemv.inl"
#include "fflas_freivalds.inl"
#include "fflas_fger.inl"
#include "fflas_fsyrk.inl"
#include "fflas_fsyrk_strassen.inl"
#include "fflas_fsyr2k.inl"
#include "fflas_ftrsm.inl"
#include "fflas_pftsm.inl"
#include "fflas_ftrmm.inl"
#include "fflas_ftrsv.inl"
#include "fflas_faxpy.inl"
#include "fflas_fdot.inl"
#include "fflas-ffpack/field/rns.h"
#include "fflas_fscal_mp.inl"
#include "fflas_freduce_mp.inl"
#include "fflas-ffpack/fflas/fflas_fger_mp.inl"
#include "fflas_fgemm/fgemm_classical_mp.inl"
#include "fflas_ftrsm_mp.inl"
#include "fflas_fgemv_mp.inl"
#include "fflas-ffpack/field/rns.inl"
#include "fflas-ffpack/paladin/fflas_plevel1.h"
#include "fflas_sparse.h"
#include "fflas-ffpack/checkers/checkers_fflas.inl"
```

## Macros

- `#define WINOTHRESHOLD __FFLASFFPACK_WINOTHRESHOLD`
- `#define DOUBLE_TO_FLOAT_CROSSOVER 800`

*Thresholds determining which floating point representation to use, depending on the cardinality of the finite field.*

## 13.67.1 Detailed Description

### Finite Field Linear Algebra Subroutines

#### Author

Clément Pernet.

## 13.67.2 Macro Definition Documentation

### 13.67.2.1 WINOTHRESHOLD

```
#define WINOTHRESHOLD __FFLASFFPACK_WINOTHRESHOLD
```

### 13.67.2.2 DOUBLE\_TO\_FLOAT\_CROSSOVER

```
#define DOUBLE_TO_FLOAT_CROSSOVER 800
```

Thresholds determining which floating point representation to use, depending on the cardinality of the finite field. This is only used when the element representation is not a floating point type.

**Bug** to be benchmarked.



## 13.68 fflas\_bounds.inl File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/utils/flimits.h"
#include <givaro/udl.h>
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
```

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::Protected](#)

### Macros

- `#define` [\\_\\_FFLASFFPACK\\_fflas\\_bounds\\_INL](#)
- `#define` [FFLAS\\_INT\\_TYPE](#) `uint64_t`

### Functions

- `template<class Field >`  
`double computeFactorClassic (const Field &F)`
- `template<> double computeFactorClassic (const Givaro::ModularBalanced< double > &F)`
- `template<> double computeFactorClassic (const Givaro::ModularBalanced< float > &F)`
- `template<class Field >`  
`size_t DotProdBoundClassic (const Field &F, const typename Field::Element &beta)`
- `Givaro::Integer InfNorm (const size_t M, const size_t N, const Givaro::Integer *A, const size_t lda)`
- `template<class Field >`  
`size_t TRSMBound (const Field &)`  
*TRSMBound.*
- `template<class Element >`  
`size_t TRSMBound (const Givaro::Modular< Element > &F)`  
*Specialization for positive modular representation over double Computes nmax s.t.*
- `template<class Element >`  
`size_t TRSMBound (const Givaro::ModularBalanced< Element > &F)`  
*Specialization for balanced modular representation over double.*

### 13.68.1 Macro Definition Documentation

#### 13.68.1.1 [\\_\\_FFLASFFPACK\\_fflas\\_bounds\\_INL](#)

```
#define __FFLASFFPACK_fflas_bounds_INL
```

#### 13.68.1.2 [FFLAS\\_INT\\_TYPE](#)

```
#define FFLAS_INT_TYPE uint64_t
```

## 13.69 fflas\_enum.h File Reference

```
#include <algorithm>
```

### Data Structures

- class [AreEqual](#)< X, Y >
- class [AreEqual](#)< X, X >

## Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::Protected](#)

## Enumerations

- enum [FFLAS\\_ORDER](#) { [FflasRowMajor](#) =101 , [FflasColMajor](#) =102 }  
*Storage by row or col ?*
- enum [FFLAS\\_TRANSPOSE](#) { [FflasNoTrans](#) = 111 , [FflasTrans](#) = 112 }  
*Is matrix transposed ?*
- enum [FFLAS\\_UPLO](#) { [FflasUpper](#) = 121 , [FflasLower](#) = 122 , [FflasLeftTri](#) = 123 , [FflasRightTri](#) = 124 }  
*Is triangular matrix's shape upper ?*
- enum [FFLAS\\_DIAG](#) { [FflasNonUnit](#) = 131 , [FflasUnit](#) = 132 }  
*Is the triangular matrix implicitly unit diagonal ?*
- enum [FFLAS\\_SIDE](#) { [FflasLeft](#) = 141 , [FflasRight](#) = 142 }  
*On what side ?*
- enum [FFLAS\\_BASE](#) { [FflasDouble](#) = 151 , [FflasFloat](#) = 152 , [FflasGeneric](#) = 153 }  
*FFLAS\_BASE determines the type of the element representation for Matrix Mult kernel.*

## Functions

- template<class T >  
const T & [min3](#) (const T &m, const T &n, const T &k)
- template<class T >  
const T & [max3](#) (const T &m, const T &n, const T &k)
- template<class T >  
const T & [min4](#) (const T &m, const T &n, const T &k, const T &l)
- template<class T >  
const T & [max4](#) (const T &m, const T &n, const T &k, const T &l)

## 13.70 fflas\_fadd.h File Reference

```
#include "fflas-ffpack/fflas/fflas_simd.h"
#include "fflas_fadd.inl"
```

## Data Structures

- struct [support\\_simd\\_add< T >](#)

## Namespaces

- namespace [FFLAS](#)

## Functions

- template<class [Field](#) >  
void [fadd](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t inca, typename [Field::ConstElement\\_ptr](#) B, const size\_t incb, typename [Field::Element\\_ptr](#) C, const size\_t incc)
- template<class [Field](#) >  
void [faddin](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) B, const size\_t incb, typename [Field::Element\\_ptr](#) C, const size\_t incc)
- template<class [Field](#) >  
void [fsub](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t inca, typename [Field::ConstElement\\_ptr](#) B, const size\_t incb, typename [Field::Element\\_ptr](#) C, const size\_t incc)

- `template<class Field >`  
`void fsubin (const Field &F, const size_t N, typename Field::ConstElement_ptr B, const size_t incb, typename Field::Element_ptr C, const size_t incc)`
- `template<class Field >`  
`void fadd (const Field &F, const size_t N, typename Field::ConstElement_ptr A, const size_t inca, const typename Field::Element alpha, typename Field::ConstElement_ptr B, const size_t incb, typename Field::Element_ptr C, const size_t incc)`
- `template<class Field >`  
`void pfadd (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc, const size_t numths)`
- `template<class Field >`  
`void pfsub (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc, const size_t numths)`
- `template<class Field >`  
`void pfaddin (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc, size_t numths)`
- `template<class Field >`  
`void pfsubin (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc, size_t numths)`
- `template<class Field >`  
`void fadd (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc)`  
*fadd : matrix addition.*
- `template<class Field >`  
`void fsub (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc)`  
*fsub : matrix subtraction.*
- `template<class Field >`  
`void faddin (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc)`  
*faddin*
- `template<class Field >`  
`void faddin (const Field &F, const FFLAS_UPLO uplo, const size_t N, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc)`  
*fadding for symmetric matrices*
- `template<class Field >`  
`void fsubin (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc)`  
*fsubin  $C = C - B$*
- `template<class Field >`  
`void fadd (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element alpha, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc)`  
*fadd : matrix addition with scaling.*

## 13.71 fflas\_fadd.inl File Reference

```
#include "fflas-ffpack/fflas/fflas_simd.h"
```

### Namespaces

- namespace `FFLAS`
- namespace `FFLAS::vectorised`
- namespace `FFLAS::details`

## Macros

- `#define __FFLASFFPACK_fadd_INL`

## Functions

- `template<class SimdT, class Element, bool positive>`  
`std::enable_if< is_simd< SimdT >::value, void >::type VEC_ADD (SimdT &C, SimdT &A, SimdT &B, SimdT &Q, SimdT &T, SimdT &P, SimdT &NEGP, SimdT &MIN, SimdT &MAX)`
- `template<bool positive, class Element, class T1, class T2 >`  
`std::enable_if< FFLAS::support_simd_add< Element >::value, void >::type addp (Element *T, const Element *TA, const Element *TB, size_t n, Element p, T1 min_, T2 max_)`
- `template<class SimdT, class Element, bool positive>`  
`std::enable_if< is_simd< SimdT >::value, void >::type VEC_SUB (SimdT &C, SimdT &A, SimdT &B, SimdT &Q, SimdT &T, SimdT &P, SimdT &NEGP, SimdT &MIN, SimdT &MAX)`
- `template<bool positive, class Element, class T1, class T2 >`  
`std::enable_if< FFLAS::support_simd_add< Element >::value, void >::type subp (Element *T, const Element *TA, const Element *TB, const size_t n, const Element p, const T1 min_, const T2 max_)`
- `template<class Element >`  
`std::enable_if< FFLAS::support_simd_add< Element >::value, void >::type add (Element *T, const Element *TA, const Element *TB, size_t n)`
- `template<class Element >`  
`std::enable_if< FFLAS::support_simd_add< Element >::value, void >::type sub (Element *T, const Element *TA, const Element *TB, size_t n)`
- `template<class Field, bool ADD>`  
`std::enable_if< FFLAS::support_simd_add< typenameField::Element >::value, void >::type fadd (const Field &F, const size_t N, typename Field::ConstElement_ptr A, const size_t inca, typename Field::ConstElement_ptr B, const size_t incb, typename Field::Element_ptr C, const size_t incc, FieldCategories::ModularTag)`
- `template<class Field, bool ADD>`  
`std::enable_if< !FFLAS::support_simd_add< typenameField::Element >::value, void >::type fadd (const Field &F, const size_t N, typename Field::ConstElement_ptr A, const size_t inca, typename Field::ConstElement_ptr B, const size_t incb, typename Field::Element_ptr C, const size_t incc, FieldCategories::ModularTag)`
- `template<class Field, bool ADD>`  
`void fadd (const Field &F, const size_t N, typename Field::ConstElement_ptr A, const size_t inca, typename Field::ConstElement_ptr B, const size_t incb, typename Field::Element_ptr C, const size_t incc, FieldCategories::GenericTag)`
- `template<class Field, bool ADD>`  
`std::enable_if< !FFLAS::support_simd_add< typenameField::Element >::value, void >::type fadd (const Field &F, const size_t N, typename Field::ConstElement_ptr A, const size_t inca, typename Field::ConstElement_ptr B, const size_t incb, typename Field::Element_ptr C, const size_t incc, FieldCategories::UnparametricTag)`
- `template<class Field, bool ADD>`  
`std::enable_if< FFLAS::support_simd_add< typenameField::Element >::value, void >::type fadd (const Field &F, const size_t N, typename Field::ConstElement_ptr A, const size_t inca, typename Field::ConstElement_ptr B, const size_t incb, typename Field::Element_ptr C, const size_t incc, FieldCategories::UnparametricTag)`

## 13.71.1 Macro Definition Documentation

### 13.71.1.1 \_\_FFLASFFPACK\_fadd\_INL

```
#define __FFLASFFPACK_fadd_INL
```

## 13.72 fflas\_fassign.h File Reference

```
#include "fflas_fassign.inl"
```

## 13.73 fflas\_fassign.inl File Reference

```
#include <string.h>
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
#include <givaro/zring.h>
#include "fflas-ffpack/utils/debug.h"
```

### Namespaces

- namespace [FFLAS](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fassign\\_INL](#)

### Functions

- template<class [Field](#) >  
void [fassign](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) Y, const size\_t incY, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
 $fassign : x \leftarrow y.$
- template<> void [fassign](#) (const Givaro::Modular< float > &F, const size\_t N, const float \*Y, const size\_t incY, float \*X, const size\_t incX)
- template<> void [fassign](#) (const Givaro::ModularBalanced< float > &F, const size\_t N, const float \*Y, const size\_t incY, float \*X, const size\_t incX)
- template<> void [fassign](#) (const Givaro::ZRing< float > &F, const size\_t N, const float \*Y, const size\_t incY, float \*X, const size\_t incX)
- template<> void [fassign](#) (const Givaro::Modular< double > &F, const size\_t N, const double \*Y, const size\_t incY, double \*X, const size\_t incX)
- template<> void [fassign](#) (const Givaro::ModularBalanced< double > &F, const size\_t N, const double \*Y, const size\_t incY, double \*X, const size\_t incX)
- template<> void [fassign](#) (const Givaro::ZRing< double > &F, const size\_t N, const double \*Y, const size\_t incY, double \*X, const size\_t incX)
- template<class [Field](#) >  
void [fassign](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
 $fassign : A \leftarrow B.$

### 13.73.1 Macro Definition Documentation

#### 13.73.1.1 \_\_FFLASFFPACK\_fassign\_INL

```
#define __FFLASFFPACK_fassign_INL
```

## 13.74 fflas\_faxpy.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::vectorised](#)
- namespace [FFLAS::vectorised::unswitch](#)
- namespace [FFLAS::details](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_faxpy\\_INL](#)

## Functions

- `template<class Field >`  
`std::enable_if<!FFLAS::support_simd_mod< typenameField::Element >::value &&FFLAS::support_fast_mod<`  
`typenameField::Element >::value, void >::type axpyp (const Field &F, const typename Field::Element a,`  
`typename Field::ConstElement_ptr X, typename Field::Element_ptr Y, const size_t n, HelperMod< Field >`  
`&H)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type axpyp`  
`(const Field &F, const typename Field::Element a, typename Field::ConstElement_ptr X, typename`  
`Field::Element_ptr Y, const size_t n, const size_t incX, const size_t incY, HelperMod< Field > &H)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type axpyp`  
`(const Field &F, const typename Field::Element a, typename Field::ConstElement_ptr X, typename`  
`Field::Element_ptr Y, const size_t n)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type axpyp`  
`(const Field &F, const typename Field::Element a, typename Field::ConstElement_ptr X, typename`  
`Field::Element_ptr Y, const size_t n, const size_t incX, const size_t incY)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type faxpy (const`  
`Field &F, const size_t N, const typename Field::Element a, typename Field::ConstElement_ptr X, const size_t`  
`incX, typename Field::Element_ptr Y, const size_t incY, FieldCategories::ModularTag)`
- `template<class Field, class FC >`  
`void faxpy (const Field &F, const size_t N, const typename Field::Element a, typename Field::ConstElement_ptr`  
`X, const size_t incX, typename Field::Element_ptr Y, const size_t incY, FC)`
- `template<class Field >`  
`void faxpy (const Field &F, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr`  
`X, const size_t incX, typename Field::Element_ptr Y, const size_t incY)`  

$$faxpy : y \leftarrow \alpha \cdot x + y.$$
- `template<> void faxpy (const Givaro::DoubleDomain &, const size_t N, const Givaro::DoubleDomain::Element a,`  
`Givaro::DoubleDomain::ConstElement_ptr x, const size_t incx, Givaro::DoubleDomain::Element_ptr y,`  
`const size_t incy)`
- `template<> void faxpy (const Givaro::FloatDomain &, const size_t N, const Givaro::FloatDomain::Element`  
`a, Givaro::FloatDomain::ConstElement_ptr x, const size_t incx, Givaro::FloatDomain::Element_ptr y, const`  
`size_t incy)`
- `template<class Field >`  
`void faxpy (const Field &F, const size_t m, const size_t n, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr X, const size_t idx, typename Field::Element_ptr Y, const size_t ldy)`  

$$faxpy : y \leftarrow \alpha \cdot x + y.$$

## 13.74.1 Macro Definition Documentation

### 13.74.1.1 \_\_FFLASFFPACK\_faxpy\_INL

```
#define __FFLASFFPACK_faxpy_INL
```

## 13.75 fflas\_fdot.inl File Reference

```
#include "fflas-ffpack/fflas/fflas_helpers.inl"
```

## Namespaces

- namespace `FFLAS`

## Macros

- `#define __FFLASFFPACK_fdot_INL`

## Functions

- `template<class Field >`  
`Field::Element fdot` (const `Field` &F, const `size_t` N, typename `Field::ConstElement_ptr` x, const `size_t` incx, typename `Field::ConstElement_ptr` y, const `size_t` incy, `ModeCategories::DefaultTag` &MT)
- `template<class Field >`  
`Field::Element fdot` (const `Field` &F, const `size_t` N, typename `Field::ConstElement_ptr` x, const `size_t` incx, typename `Field::ConstElement_ptr` y, const `size_t` incy, `ModeCategories::DelayedTag` &MT)
- `template<> Givaro::DoubleDomain::Element fdot` (const `Givaro::DoubleDomain` &, const `size_t` N, `Givaro::DoubleDomain::ConstElement_ptr` x, const `size_t` incx, `Givaro::DoubleDomain::ConstElement_ptr` y, const `size_t` incy, `ModeCategories::DefaultTag` &MT)
- `template<> Givaro::FloatDomain::Element fdot` (const `Givaro::FloatDomain` &, const `size_t` N, `Givaro::FloatDomain::ConstElement_ptr` x, const `size_t` incx, `Givaro::FloatDomain::ConstElement_ptr` y, const `size_t` incy, `ModeCategories::DefaultTag` &MT)
- `template<class Field, class T >`  
`Field::Element fdot` (const `Field` &F, const `size_t` N, typename `Field::ConstElement_ptr` x, const `size_t` incx, typename `Field::ConstElement_ptr` y, const `size_t` incy, `ModeCategories::ConvertTo`< T > &MT)
- `template<class Field >`  
`Field::Element fdot` (const `Field` &F, const `size_t` N, typename `Field::ConstElement_ptr` x, const `size_t` incx, typename `Field::ConstElement_ptr` y, const `size_t` incy, `ModeCategories::DefaultBoundedTag` &dbt)
- `template<class Field >`  
`Field::Element fdot` (const `Field` &F, const `size_t` N, typename `Field::ConstElement_ptr` x, const `size_t` incx, typename `Field::ConstElement_ptr` y, const `size_t` incy, const `ParSeqHelper::Sequential` seq)
- `template<class Field >`  
`Field::Element fdot` (const `Field` &F, const `size_t` N, typename `Field::ConstElement_ptr` X, const `size_t` incX, typename `Field::ConstElement_ptr` Y, const `size_t` incY)  

$$fdot: \text{dot product } x^T y.$$

## 13.75.1 Macro Definition Documentation

### 13.75.1.1 \_\_FFLASFFPACK\_fdot\_INL

```
#define __FFLASFFPACK_fdot_INL
```

## 13.76 fflas\_fgemm.inl File Reference

```
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/debug.h"
#include "fflas_fgemm/fgemm_classical.inl"
#include "fflas_fgemm/fgemm_winograd.inl"
```

## Namespaces

- namespace `FFLAS`
- namespace `FFLAS::Protected`

## Macros

- `#define __FFLASFFPACK_fgemm_INL`

## Functions

- `template<class NewField , class Field , class FieldMode >`  
`Field::Element_ptr fgemm_convert` (const `Field` &F, const `FFLAS_TRANSPOSE` ta, const `FFLAS_TRANSPOSE` tb, const `size_t` m, const `size_t` n, const `size_t` k, const `typename` `Field::Element` alpha, `typename` `Field::ConstElement_ptr` A, const `size_t` lda, `typename` `Field::ConstElement_ptr` B, const `size_t` ldb, const `typename` `Field::Element` beta, `typename` `Field::Element_ptr` C, const `size_t` ldc, `MMHelper`< `Field`, `MMHelperAlgo::Winograd`, `FieldMode` > &H)
- `template<class Field , class Element , class AlgoT , class ParSeqTrait >`  
`bool NeedPreAddReduction` (`Element` &Outmin, `Element` &Outmax, `Element` &Op1min, `Element` &Op1max, `Element` &Op2min, `Element` &Op2max, `MMHelper`< `Field`, `AlgoT`, `ModeCategories::LazyTag`, `ParSeqTrait` > &WH)
- `template<class Field , class Element , class AlgoT , class ModeT , class ParSeqTrait >`  
`bool NeedPreAddReduction` (`Element` &Outmin, `Element` &Outmax, `Element` &Op1min, `Element` &Op1max, `Element` &Op2min, `Element` &Op2max, `MMHelper`< `Field`, `AlgoT`, `ModeT`, `ParSeqTrait` > &WH)
- `template<class Field , class Element , class AlgoT , class ParSeqTrait >`  
`bool NeedPreSubReduction` (`Element` &Outmin, `Element` &Outmax, `Element` &Op1min, `Element` &Op1max, `Element` &Op2min, `Element` &Op2max, `MMHelper`< `Field`, `AlgoT`, `ModeCategories::LazyTag`, `ParSeqTrait` > &WH)
- `template<class Field , class Element , class AlgoT , class ModeT , class ParSeqTrait >`  
`bool NeedPreSubReduction` (`Element` &Outmin, `Element` &Outmax, `Element` &Op1min, `Element` &Op1max, `Element` &Op2min, `Element` &Op2max, `MMHelper`< `Field`, `AlgoT`, `ModeT`, `ParSeqTrait` > &WH)
- `template<class Field , class Element , class AlgoT , class ParSeqTrait >`  
`bool NeedDoublePreAddReduction` (`Element` &Outmin, `Element` &Outmax, `Element` &Op1min, `Element` &Op1max, `Element` &Op2min, `Element` &Op2max, `Element` beta, `MMHelper`< `Field`, `AlgoT`, `ModeCategories::LazyTag`, `ParSeqTrait` > &WH)
- `template<class Field , class Element , class AlgoT , class ModeT , class ParSeqTrait >`  
`bool NeedDoublePreAddReduction` (`Element` &Outmin, `Element` &Outmax, `Element` &Op1min, `Element` &Op1max, `Element` &Op2min, `Element` &Op2max, `Element` beta, `MMHelper`< `Field`, `AlgoT`, `ModeT`, `ParSeqTrait` > &WH)
- `template<class Field , class AlgoT , class ParSeqTrait >`  
`void ScalAndReduce` (const `Field` &F, const `size_t` N, const `typename` `Field::Element` alpha, `typename` `Field::Element_ptr` X, const `size_t` incX, const `MMHelper`< `Field`, `AlgoT`, `ModeCategories::LazyTag`, `ParSeqTrait` > &H)
- `template<class Field , class AlgoT , class ParSeqTrait >`  
`void ScalAndReduce` (const `Field` &F, const `size_t` M, const `size_t` N, const `typename` `Field::Element` alpha, `typename` `Field::Element_ptr` A, const `size_t` lda, const `MMHelper`< `Field`, `AlgoT`, `ModeCategories::LazyTag`, `ParSeqTrait` > &H)
- `template<class Field >`  
`Field::Element_ptr fgemm` (const `Field` &F, const `FFLAS_TRANSPOSE` ta, const `FFLAS_TRANSPOSE` tb, const `size_t` m, const `size_t` n, const `size_t` k, const `typename` `Field::Element` alpha, `typename` `Field::ConstElement_ptr` A, const `size_t` lda, `typename` `Field::ConstElement_ptr` B, const `size_t` ldb, const `typename` `Field::Element` beta, `typename` `Field::Element_ptr` C, const `size_t` ldc, `MMHelper`< `Field`, `MMHelperAlgo::Winograd`, `ModeCategories::ConvertTo`< `ElementCategories::MachineFloatTag` >, `ParSeqHelper::Sequential` > &H)
- `template<typename Field >`  
`Field::Element_ptr fgemm` (const `Field` &F, const `FFLAS_TRANSPOSE` ta, const `FFLAS_TRANSPOSE` tb, const `size_t` m, const `size_t` n, const `size_t` k, const `typename` `Field::Element` alpha, `typename` `Field::ConstElement_ptr` A, const `size_t` lda, `typename` `Field::ConstElement_ptr` B, const `size_t` ldb, const `typename` `Field::Element` beta, `typename` `Field::Element_ptr` C, const `size_t` ldc, const `ParSeqHelper::Sequential` seq)
- `template<typename Field , class Cut , class Param >`  
`Field::Element_ptr fgemm` (const `Field` &F, const `FFLAS_TRANSPOSE` ta, const `FFLAS_TRANSPOSE` tb, const `size_t` m, const `size_t` n, const `size_t` k, const `typename` `Field::Element` alpha, `typename` `Field::ConstElement_ptr` A, const `size_t` lda, `typename` `Field::ConstElement_ptr` B, const `size_t` ldb, const `typename` `Field::Element` beta, `typename` `Field::Element_ptr` C, const `size_t` ldc, const `ParSeqHelper::Parallel`< `Cut`, `Param` > par)



- `template<typename Field >`  
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc)`  
*fgemm: Field GENeral Matrix Multiply.*
- `template<typename Field , class ModeT , class ParSeq >`  
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Auto, ModeT, ParSeq > &H)`
- `template<class Field >`  
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, ModeCategories::DelayedTag, ParSeqHelper::Sequential > &H)`
- `template<class Field >`  
`Field::Element_ptr fsquare (const Field &F, const FFLAS_TRANSPOSE ta, const size_t n, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc)`  
*fsquare: Squares a matrix.*
- `template<class Field >`  
`Field::Element_ptr fsquareCommon (const Field &F, const FFLAS_TRANSPOSE ta, const size_t n, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc)`
- `template<> double * fsquare (const Givaro::ModularBalanced< double > &F, const FFLAS_TRANSPOSE ta, const size_t n, const double alpha, const double *A, const size_t lda, const double beta, double *C, const size_t ldc)`
- `template<> float * fsquare (const Givaro::ModularBalanced< float > &F, const FFLAS_TRANSPOSE ta, const size_t n, const float alpha, const float *A, const size_t lda, const float beta, float *C, const size_t ldc)`
- `template<> double * fsquare (const Givaro::Modular< double > &F, const FFLAS_TRANSPOSE ta, const size_t n, const double alpha, const double *A, const size_t lda, const double beta, double *C, const size_t ldc)`
- `template<> float * fsquare (const Givaro::Modular< float > &F, const FFLAS_TRANSPOSE ta, const size_t n, const float alpha, const float *A, const size_t lda, const float beta, float *C, const size_t ldc)`

## 13.76.1 Macro Definition Documentation

### 13.76.1.1 \_\_FFLASFFPACK\_fgemm\_INL

```
#define __FFLASFFPACK_fgemm_INL
```

## 13.77 fgemm\_classical.inl File Reference

## 13.78 fgemm\_classical\_mp.inl File Reference

matrix multiplication with multiprecision input (either over Z or over Z/pZ)

```
#include <givaro/modular-integer.h>
#include <givaro/zring.h>
#include "fflas-ffpack/field/rns-double.h"
#include "fflas-ffpack/field/rns-integer.h"
#include "fflas-ffpack/field/rns-integer-mod.h"
#include "fflas-ffpack/field/field-traits.h"
#include "fflas-ffpack/fflas/fflas_helpers.inl"
```

```
#include "fflas-ffpack/fflas/fflas_bounds.inl"
```

## Data Structures

- struct [MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeqTrait >](#)
- struct [MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >](#)
- struct [MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >](#)

## Namespaces

- namespace [FFLAS](#)

## Macros

- #define [\\_\\_FFPACK\\_fgemm\\_classical\\_INL](#)

## Functions

- template<typename [RNS](#) , typename [ParSeqTrait](#) >  
[FFPACK::RNSInteger< RNS >::Element\\_ptr fgemm](#) (const [FFPACK::RNSInteger< RNS >](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t m, const size\_t n, const size\_t k, const typename [FFPACK::RNSInteger< RNS >::Element](#) alpha, typename [FFPACK::RNSInteger< RNS >::ConstElement\\_ptr](#) Ad, const size\_t lda, typename [FFPACK::RNSInteger< RNS >::ConstElement\\_ptr](#) Bd, const size\_t ldb, const typename [FFPACK::RNSInteger< RNS >::Element](#) beta, typename [FFPACK::RNSInteger< RNS >::Element\\_ptr](#) Cd, const size\_t ldc, [MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag, ParSeqHelper::Compose< ParSeqHelper::Sequential, ParSeqTrait > >](#) &H)
- template<typename [RNS](#) >  
[FFPACK::RNSInteger< RNS >::Element\\_ptr fgemm](#) (const [FFPACK::RNSInteger< RNS >](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t m, const size\_t n, const size\_t k, const typename [FFPACK::RNSInteger< RNS >::Element](#) alpha, typename [FFPACK::RNSInteger< RNS >::ConstElement\\_ptr](#) Ad, const size\_t lda, typename [FFPACK::RNSInteger< RNS >::ConstElement\\_ptr](#) Bd, const size\_t ldb, const typename [FFPACK::RNSInteger< RNS >::Element](#) beta, typename [FFPACK::RNSInteger< RNS >::Element\\_ptr](#) Cd, const size\_t ldc, [MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag, ParSeqHelper::Sequential >](#) &H)
- template<typename [RNS](#) , typename [ParSeqTrait](#) >  
[FFPACK::RNSInteger< RNS >::Element\\_ptr fgemm](#) (const [FFPACK::RNSInteger< RNS >](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t m, const size\_t n, const size\_t k, const typename [FFPACK::RNSInteger< RNS >::Element](#) alpha, typename [FFPACK::RNSInteger< RNS >::ConstElement\\_ptr](#) Ad, const size\_t lda, typename [FFPACK::RNSInteger< RNS >::ConstElement\\_ptr](#) Bd, const size\_t ldb, const typename [FFPACK::RNSInteger< RNS >::Element](#) beta, typename [FFPACK::RNSInteger< RNS >::Element\\_ptr](#) Cd, const size\_t ldc, [MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag, ParSeqHelper::Compose< ParSeqHelper::Parallel< CuttingStrategy::RNSModulus, StrategyParameter::Threads >, ParSeqTrait > >](#) &H)
- template<typename [RNS](#) , typename [Cut](#) , typename [Param](#) >  
[FFPACK::RNSInteger< RNS >::Element\\_ptr fgemm](#) (const [FFPACK::RNSInteger< RNS >](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t m, const size\_t n, const size\_t k, const typename [FFPACK::RNSInteger< RNS >::Element](#) alpha, typename [FFPACK::RNSInteger< RNS >::ConstElement\\_ptr](#) Ad, const size\_t lda, typename [FFPACK::RNSInteger< RNS >::ConstElement\\_ptr](#) Bd, const size\_t ldb, const typename [FFPACK::RNSInteger< RNS >::Element](#) beta, typename [FFPACK::RNSInteger< RNS >::Element\\_ptr](#) Cd, const size\_t ldc, [MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag, ParSeqHelper::Parallel< Cut, Param > >](#) &H)
- template<class [ParSeq](#) >  
[Givaro::Integer \\* fgemm](#) (const [Givaro::ZRing< Givaro::Integer >](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t m, const size\_t n, const size\_t k, const [Givaro::Integer](#) alpha, const [Givaro::Integer](#) \*A, const size\_t lda, const [Givaro::Integer](#) \*B, const size\_t ldb, [Givaro::Integer](#) beta,

- Givaro::Integer \*C, const size\_t ldc, MMHelper< Givaro::ZRing< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq > &H)
- template<typename RNS, class ModeT >  
RNS::Element\_ptr fgemm (const FFPACK::RNSInteger< RNS > &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE tb, const size\_t m, const size\_t n, const size\_t k, const typename RNS::Element alpha, typename RNS::ConstElement\_ptr Ad, const size\_t lda, typename RNS::ConstElement\_ptr Bd, const size\_t ldb, const typename RNS::Element beta, typename RNS::Element\_ptr Cd, const size\_t ldc, MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Winograd, ModeT, ParSeqHelper::Sequential > &H)
  - template<typename RNS >  
RNS::Element\_ptr fgemm (const FFPACK::RNSIntegerMod< RNS > &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE tb, const size\_t m, const size\_t n, const size\_t k, const typename RNS::Element alpha, typename RNS::ConstElement\_ptr Ad, const size\_t lda, typename RNS::ConstElement\_ptr Bd, const size\_t ldb, const typename RNS::Element beta, typename RNS::Element\_ptr Cd, const size\_t ldc, MMHelper< FFPACK::RNSIntegerMod< RNS >, MMHelperAlgo::Winograd > &H)
  - Givaro::Integer \* fgemm (const Givaro::Modular< Givaro::Integer > &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE tb, const size\_t m, const size\_t n, const size\_t k, const Givaro::Integer alpha, const Givaro::Integer \*A, const size\_t lda, const Givaro::Integer \*B, const size\_t ldb, const Givaro::Integer beta, Givaro::Integer \*C, const size\_t ldc, MMHelper< Givaro::Modular< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::RNSElementTag > > &H)
  - template<class ParSeq >  
Givaro::Integer \* fgemm (const Givaro::Modular< Givaro::Integer > &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE tb, const size\_t m, const size\_t n, const size\_t k, const Givaro::Integer alpha, const Givaro::Integer \*A, const size\_t lda, const Givaro::Integer \*B, const size\_t ldb, const Givaro::Integer beta, Givaro::Integer \*C, const size\_t ldc, MMHelper< Givaro::Modular< Givaro::Integer >, MMHelperAlgo::Auto, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq > &H)
  - template<size\_t K1, size\_t K2, class ParSeq >  
RecInt::ruint< K1 > \* fgemm (const Givaro::Modular< RecInt::ruint< K1 >, RecInt::ruint< K2 > > &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE tb, const size\_t m, const size\_t n, const size\_t k, const RecInt::ruint< K1 > alpha, const RecInt::ruint< K1 > \*A, const size\_t lda, const RecInt::ruint< K1 > \*B, const size\_t ldb, RecInt::ruint< K1 > beta, RecInt::ruint< K1 > \*C, const size\_t ldc, MMHelper< Givaro::Modular< RecInt::ruint< K1 >, RecInt::ruint< K2 > >, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq > &H)

### 13.78.1 Detailed Description

matrix multiplication with multiprecision input (either over Z or over Z/pZ)

### 13.78.2 Macro Definition Documentation

#### 13.78.2.1 \_\_FFPACK\_fgemm\_classical\_INL

```
#define __FFPACK_fgemm_classical_INL
```

## 13.79 fgemm\_winograd.inl File Reference

```
#include <stdint.h>
#include <givaro/modular.h>
#include <givaro/zring.h>
#include "fgemm_classical.inl"
#include "schedule_winograd.inl"
#include "schedule_winograd_acc.inl"
#include "schedule_winograd_acc_ip.inl"
#include "schedule_winograd_ip.inl"
#include "fflas-ffpack/fflas-ffpack-config.h"
```

## Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::Protected](#)

## Macros

- `#define` [\\_\\_FFLASFFPACK\\_fflas\\_fflas\\_fgemm\\_winograd\\_INL](#)
- `#define` [NEWWINO](#)

## Functions

- `template<class Field >`  
`int WinogradThreshold (const Field &F)`  
*Computes the number of recursive levels to perform.*
- `template<> int WinogradThreshold (const Givaro::Modular< float > &F)`
- `template<> int WinogradThreshold (const Givaro::ModularBalanced< double > &F)`
- `template<> int WinogradThreshold (const Givaro::ModularBalanced< float > &F)`
- `template<class Field >`  
`int WinogradSteps (const Field &F, const size_t &m)`  
*Computes the number of recursive levels to perform.*
- `template<class Field , class FieldMode >`  
`void DynamicPeeling (const Field &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, typename Field::ConstElement\_ptr A, const size_t lda, typename Field::ConstElement\_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element\_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, FieldMode > &H, const typename MMHelper< Field, MMHelperAlgo::Winograd, FieldMode >::DelayedField::Element Cmin, const typename MMHelper< Field, MMHelperAlgo::Winograd, FieldMode >::DelayedField::Element Cmax)`
- `template<class Field , class FieldMode >`  
`void DynamicPeeling2 (const Field &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, typename Field::ConstElement\_ptr A, const size_t lda, typename Field::ConstElement\_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element\_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, FieldMode > &H, const typename MMHelper< Field, MMHelperAlgo::Winograd, FieldMode >::DelayedField::Element Cmin, const typename MMHelper< Field, MMHelperAlgo::Winograd, FieldMode >::DelayedField::Element Cmax)`
- `template<class Field , class FieldMode >`  
`void WinogradCalc (const Field &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE tb, const size_t nr, const size_t kr, const typename Field::Element alpha, typename Field::ConstElement\_ptr A, const size_t lda, typename Field::ConstElement\_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element\_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, FieldMode > &H)`
- `template<class Field , class ModeT >`  
`Field::Element\_ptr fgemm (const Field &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement\_ptr A, const size_t lda, typename Field::ConstElement\_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element\_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, ModeT > &H)`
- `template<class Field , class ModeT , class Cut , class Param >`  
`Field::Element\_ptr fgemm (const Field &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement\_ptr A, const size_t lda, typename Field::ConstElement\_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element\_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::WinogradPar, ModeT, ParSeqHelper::Parallel< Cut, Param > > &H)`

## 13.79.1 Macro Definition Documentation

### 13.79.1.1 \_\_FFLASFFPACK\_fflas\_fflas\_fgemm\_winograd\_INL

```
#define __FFLASFFPACK_fflas_fflas_fgemm_winograd_INL
```

### 13.79.1.2 NEWWINO

```
#define NEWWINO
```

## 13.80 matmul.doxy File Reference

## 13.81 schedule\_bini.inl File Reference

Bini implementation.

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::BLAS3](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fgemm\\_bini\\_INL](#)

### Functions

- template<class [Field](#) >  
void [Bini](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t mr, const size\_t nr, const size\_t kr, const typename [Field::Element](#) alpha, const typename [Field::Element\\_ptr](#) A, const size\_t lda, const typename [Field::Element\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, const typename [Field::Element\\_ptr](#) C, const size\_t ldc, const size\_t kmax, const size\_t w, const [FFLAS\\_BASE](#) base, const size\_t rec\_level)

### 13.81.1 Detailed Description

Bini implementation.

### 13.81.2 Macro Definition Documentation

#### 13.81.2.1 \_\_FFLASFFPACK\_fgemm\_bini\_INL

```
#define __FFLASFFPACK_fgemm_bini_INL
```

## 13.82 schedule\_winograd.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::BLAS3](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fgemm\\_winograd\\_INL](#)

## Functions

- `template<class Field , class FieldTrait , class Strat , class Param >`  
`Field::Element_ptr WinoPar (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE`  
`tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb,`  
`const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field,`  
`MMHelperAlgo::WinogradPar, FieldTrait, ParSeqHelper::Parallel< Strat, Param > > &WH)`
- `template<class Field , class FieldTrait >`  
`void Winograd (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t`  
`mr, const size_t nr, const size_t kr, const typename Field::Element alpha, typename Field::ConstElement_ptr`  
`A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element`  
`beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, Field↵`  
`Trait > &WH)`

## 13.82.1 Macro Definition Documentation

### 13.82.1.1 \_\_FFLASFFPACK\_fgemm\_winograd\_INL

```
#define __FFLASFFPACK_fgemm_winograd_INL
```

## 13.83 schedule\_winograd\_acc.inl File Reference

### Namespaces

- namespace `FFLAS`
- namespace `FFLAS::BLAS3`

### Macros

- `#define __FFLASFFPACK_fgemm_winograd_acc_INL`

### Functions

- `template<class Field , class FieldTrait >`  
`void WinogradAcc_3_23 (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE`  
`tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb,`  
`const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field,`  
`MMHelperAlgo::Winograd, FieldTrait > &WH)`
- `template<class Field , class FieldTrait >`  
`void WinogradAcc_3_21 (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE`  
`tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb,`  
`const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field,`  
`MMHelperAlgo::Winograd, FieldTrait > &WH)`
- `template<class Field , class FieldTrait >`  
`void WinogradAcc_2_24 (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE`  
`tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, const type-`  
`name Field::Element_ptr A, const size_t lda, const typename Field::Element_ptr B, const size_t ldb,`  
`const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field,`  
`MMHelperAlgo::Winograd, FieldTrait > &WH)`
- `template<class Field , class FieldTrait >`  
`void WinogradAcc_2_27 (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE`  
`tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, const type-`  
`name Field::Element_ptr A, const size_t lda, const typename Field::Element_ptr B, const size_t ldb,`  
`const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field,`  
`MMHelperAlgo::Winograd, FieldTrait > &WH)`

### 13.83.1 Macro Definition Documentation

#### 13.83.1.1 \_\_FFLASFFPACK\_fgemm\_winograd\_acc\_INL

```
#define __FFLASFFPACK_fgemm_winograd_acc_INL
```

## 13.84 schedule\_winograd\_acc\_ip.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::BLAS3](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fgemm\\_winograd\\_acc\\_ip\\_INL](#)

### Functions

- template<class [Field](#) , class FieldTrait >  
void [WinogradAcc\\_LR](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t mr, const size\_t nr, const size\_t kr, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, const [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), Field↵Trait > &WH)
- template<class [Field](#) , class FieldTrait >  
void [WinogradAcc\\_R\\_S](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t mr, const size\_t nr, const size\_t kr, const typename [Field::Element](#) alpha, const type-  
name [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) B, const size\_t ldb, const type-  
name [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, const [MMHelper](#)< [Field](#),  
[MMHelperAlgo::Winograd](#), FieldTrait > &WH)
- template<class [Field](#) , class FieldTrait >  
void [WinogradAcc\\_L\\_S](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t mr, const size\_t nr, const size\_t kr, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) A, const size\_t lda, const typename [Field::Element\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, const [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), FieldTrait > &WH)

### 13.84.1 Macro Definition Documentation

#### 13.84.1.1 \_\_FFLASFFPACK\_fgemm\_winograd\_acc\_ip\_INL

```
#define __FFLASFFPACK_fgemm_winograd_acc_ip_INL
```

## 13.85 schedule\_winograd\_ip.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::BLAS3](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fgemm\\_winograd\\_ip\\_INL](#)



## Functions

- `template<class Field , class FieldTrait >`  
`void Winograd_LR_S (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const MMHelper< Field, MMHelperAlgo::Winograd, Field↔Trait > &WH)`
- `template<class Field , class FieldTrait >`  
`void Winograd_L_S (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, const typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)`
- `template<class Field , class FieldTrait >`  
`void Winograd_R_S (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, const typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)`

### 13.85.1 Macro Definition Documentation

#### 13.85.1.1 \_\_FFLASFFPACK\_fgemv\_winograd\_ip\_INL

```
#define __FFLASFFPACK_fgemv_winograd_ip_INL
```

## 13.86 fflas\_fgemv.inl File Reference

```
#include <givaro/zring.h>
```

## Namespaces

- namespace `FFLAS`
- namespace `FFLAS::Protected`

## Macros

- `#define __FFLASFFPACK_fgemv_INL`

## Functions

- `template<typename FloatElement , class Field >`  
`Field::Element_ptr fgemv_convert (const Field &F, const FFLAS_TRANSPOSE ta, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr X, const size_t incX, const typename Field::Element beta, typename Field::Element_ptr Y, const size_t incY)`
- `template<class Field >`  
`Field::Element_ptr fgemv (const Field &F, const FFLAS_TRANSPOSE ta, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr X, const size_t incX, const typename Field::Element beta, typename Field::Element_ptr Y, const size_t incY, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::MachineFloatTag > > &H)`
- `template<class Field >`  
`Field::Element_ptr fgemv (const Field &F, const FFLAS_TRANSPOSE ta, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr X, const size_t incX, const typename Field::Element`



- beta, typename [Field::Element\\_ptr](#) Y, const size\_t incY, [MMHelper](#)< [Field](#), [MMHelperAlgo::Classic](#), [ModeCategories::DelayedTag](#) > &H)
- `template<class Field >`  
[Field::Element\\_ptr](#) fgmv (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) Y, const size\_t incY, [MMHelper](#)< [Field](#), [MMHelperAlgo::Classic](#), [ModeCategories::DefaultTag](#) > &H)
  - `template<class Field >`  
[Field::Element\\_ptr](#) fgmv (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) Y, const size\_t incY, [MMHelper](#)< [Field](#), [MMHelperAlgo::Classic](#), [ModeCategories::LazyTag](#) > &H)
  - `template<class Field >`  
[Field::Element\\_ptr](#) fgmv (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) TransA, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) Y, const size\_t incY)
- finite prime Field GEneral Matrix Vector multiplication.*
- [Givaro::ZRing](#)< int64\_t >::Element\_ptr fgmv (const [Givaro::ZRing](#)< int64\_t > &F, const [FFLAS\\_TRANSPOSE](#) ta, const size\_t M, const size\_t N, const int64\_t alpha, const int64\_t \*A, const size\_t lda, const int64\_t \*X, const size\_t incX, const int64\_t beta, const int64\_t \*Y, const size\_t incY, [MMHelper](#)< [Givaro::ZRing](#)< int64\_t >, [MMHelperAlgo::Classic](#), [ModeCategories::DefaultTag](#) > &H)
  - [Givaro::DoubleDomain::Element\\_ptr](#) fgmv (const [Givaro::DoubleDomain](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const size\_t M, const size\_t N, const [Givaro::DoubleDomain::Element](#) alpha, const [Givaro::DoubleDomain::ConstElement\\_ptr](#) A, const size\_t lda, const [Givaro::DoubleDomain::ConstElement\\_ptr](#) X, const size\_t incX, const [Givaro::DoubleDomain::Element](#) beta, [Givaro::DoubleDomain::Element\\_ptr](#) Y, const size\_t incY, [MMHelper](#)< [Givaro::DoubleDomain](#), [MMHelperAlgo::Classic](#), [ModeCategories::DefaultTag](#) > &H)
  - `template<class Field >`  
[Field::Element\\_ptr](#) fgmv (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, const typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, const typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) Y, const size\_t incY, [MMHelper](#)< [Field](#), [MMHelperAlgo::Classic](#), [ModeCategories::DefaultBoundedTag](#) > &H)
  - [Givaro::FloatDomain::Element\\_ptr](#) fgmv (const [Givaro::FloatDomain](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const size\_t M, const size\_t N, const [Givaro::FloatDomain::Element](#) alpha, const [Givaro::FloatDomain::ConstElement\\_ptr](#) A, const size\_t lda, const [Givaro::FloatDomain::ConstElement\\_ptr](#) X, const size\_t incX, const [Givaro::FloatDomain::Element](#) beta, [Givaro::FloatDomain::Element\\_ptr](#) Y, const size\_t incY, [MMHelper](#)< [Givaro::FloatDomain](#), [MMHelperAlgo::Classic](#), [ModeCategories::DefaultTag](#) > &H)
  - `template<class Field , class Cut , class Param >`  
[Field::Element\\_ptr](#) fgmv (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const size\_t m, const size\_t n, const typename [Field::Element](#) alpha, const typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, const typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) Y, const size\_t incY, [ParSeqHelper::Parallel](#)< Cut, Param > &parH)
  - `template<class Field >`  
[Field::Element\\_ptr](#) fgmv (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const size\_t m, const size\_t n, const typename [Field::Element](#) alpha, const typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, const typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) Y, const size\_t incY, [ParSeqHelper::Sequential](#) &seqH)

## 13.86.1 Macro Definition Documentation

### 13.86.1.1 \_\_FFLASFFPACK\_fgmv\_INL

```
#define __FFLASFFPACK_fgmv_INL
```

## 13.87 fflas\_fgmv\_mp.inl File Reference

```
#include "fflas-ffpack/field/rns-integer-mod.h"
```

### Namespaces

- namespace [FFLAS](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fgmv\\_mp\\_INL](#)

### Functions

- [FFPACK::rns\\_double::Element\\_ptr fgmv](#) (const [FFPACK::RNSInteger](#)< [FFPACK::rns\\_double](#) > &F, const [FFLAS\\_TRANSPOSE](#) ta, const size\_t M, const size\_t N, const [FFPACK::rns\\_double::Element](#) alpha, [FFPACK::rns\\_double::ConstElement\\_ptr](#) A, const size\_t lda, [FFPACK::rns\\_double::ConstElement\\_ptr](#) X, const size\_t incX, const [FFPACK::rns\\_double::Element](#) beta, [FFPACK::rns\\_double::Element\\_ptr](#) Y, const size\_t incY, [MMHelper](#)< [FFPACK::RNSInteger](#)< [FFPACK::rns\\_double](#) >, [MMHelperAlgo::Classic](#), [ModeCategories::DefaultTag](#) > &H)
- [FFPACK::rns\\_double::Element\\_ptr fgmv](#) (const [FFPACK::RNSIntegerMod](#)< [FFPACK::rns\\_double](#) > &F, const [FFLAS\\_TRANSPOSE](#) ta, const size\_t M, const size\_t N, const [FFPACK::rns\\_double::Element](#) alpha, [FFPACK::rns\\_double::ConstElement\\_ptr](#) A, const size\_t lda, [FFPACK::rns\\_double::ConstElement\\_ptr](#) X, const size\_t incX, const [FFPACK::rns\\_double::Element](#) beta, [FFPACK::rns\\_double::Element\\_ptr](#) Y, const size\_t incY, [MMHelper](#)< [FFPACK::RNSIntegerMod](#)< [FFPACK::rns\\_double](#) >, [MMHelperAlgo::Classic](#), [ModeCategories::DefaultTag](#) > &H)
- [Givaro::Integer \\* fgmv](#) (const [Givaro::ZRing](#)< [Givaro::Integer](#) > &F, const [FFLAS\\_TRANSPOSE](#) ta, const size\_t m, const size\_t n, const [Givaro::Integer](#) alpha, [Givaro::Integer \\*A](#), const size\_t lda, [Givaro::Integer \\*X](#), const size\_t ldx, [Givaro::Integer](#) beta, [Givaro::Integer \\*Y](#), const size\_t ldy, [MMHelper](#)< [Givaro::ZRing](#)< [Givaro::Integer](#) >, [MMHelperAlgo::Classic](#), [ModeCategories::ConvertTo](#)< [ElementCategories::RNSElementTag](#) > > &H)
- [Givaro::Integer \\* fgmv](#) (const [Givaro::Modular](#)< [Givaro::Integer](#) > &F, const [FFLAS\\_TRANSPOSE](#) ta, const size\_t m, const size\_t n, const [Givaro::Integer](#) alpha, [Givaro::Integer \\*A](#), const size\_t lda, [Givaro::Integer \\*X](#), const size\_t ldx, [Givaro::Integer](#) beta, [Givaro::Integer \\*Y](#), const size\_t ldy, [MMHelper](#)< [Givaro::Modular](#)< [Givaro::Integer](#) >, [MMHelperAlgo::Classic](#), [ModeCategories::ConvertTo](#)< [ElementCategories::RNSElementTag](#) > > &H)
- [template<size\\_t K1, size\\_t K2, class ParSeq>](#) [Reclnt::ruint< K1 > \\* fgmv](#) (const [Givaro::Modular](#)< [Reclnt::ruint< K1 >](#), [Reclnt::ruint< K2 >](#) > &F, const [FFLAS\\_TRANSPOSE](#) ta, const size\_t m, const size\_t n, const [Reclnt::ruint< K1 >](#) alpha, const [Reclnt::ruint< K1 > \\*A](#), const size\_t lda, const [Reclnt::ruint< K1 > \\*X](#), const size\_t incx, [Reclnt::ruint< K1 >](#) beta, [Reclnt::ruint< K1 > \\*Y](#), const size\_t incy, [MMHelper](#)< [Givaro::Modular](#)< [Reclnt::ruint< K1 >](#), [Reclnt::ruint< K2 >](#) >, [MMHelperAlgo::Classic](#), [ModeCategories::ConvertTo](#)< [ElementCategories::RNSElementTag](#) >, [ParSeq](#) > &H)

### 13.87.1 Macro Definition Documentation

#### 13.87.1.1 \_\_FFLASFFPACK\_fgmv\_mp\_INL

```
#define __FFLASFFPACK_fgmv_mp_INL
```

## 13.88 fflas\_fger.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::Protected](#)

## Macros

- `#define __FFLASFFPACK_fger_INL`

## Functions

- `template<class Field >`  
`void fger (const Field &F, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, typename Field::Element_ptr A, const size_t lda)`  
*fger: rank one update of a general matrix*
- `template<class FloatElement, class Field >`  
`void fger_convert (const Field &F, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, typename Field::Element_ptr A, const size_t lda)`
- `template<class Field >`  
`void fger (const Field &F, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, typename Field::Element_ptr A, const size_t lda, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::MachineFloatTag > > &H)`
- `template<class Field, class AnyTag >`  
`void fger (const Field &F, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, typename Field::Element_ptr A, const size_t lda, MMHelper< Field, MMHelperAlgo::Classic, AnyTag > &H)`
- `void fger (const Givaro::DoubleDomain &F, const size_t M, const size_t N, const Givaro::DoubleDomain::Element alpha, const Givaro::DoubleDomain::ConstElement_ptr x, const size_t incx, const Givaro::DoubleDomain::ConstElement_ptr y, const size_t incy, Givaro::DoubleDomain::Element_ptr A, const size_t lda, MMHelper< Givaro::DoubleDomain, MMHelperAlgo::Classic, ModeCategories::DefaultTag > &H)`
- `template<class Field >`  
`void fger (const Field &F, const size_t M, const size_t N, const typename Field::Element alpha, const typename Field::ConstElement_ptr x, const size_t incx, const typename Field::ConstElement_ptr y, const size_t incy, typename Field::Element_ptr A, const size_t lda, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DefaultBoundedTag > &H)`
- `void fger (const Givaro::FloatDomain &F, const size_t M, const size_t N, const Givaro::FloatDomain::Element alpha, const Givaro::FloatDomain::ConstElement_ptr x, const size_t incx, const Givaro::FloatDomain::ConstElement_ptr y, const size_t incy, Givaro::FloatDomain::Element_ptr A, const size_t lda, MMHelper< Givaro::FloatDomain, MMHelperAlgo::Classic, ModeCategories::DefaultTag > &H)`
- `template<class Field >`  
`void fger (const Field &F, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, typename Field::Element_ptr A, const size_t lda, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::LazyTag > &H)`
- `template<class Field >`  
`void fger (const Field &F, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, typename Field::Element_ptr A, const size_t lda, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DelayedTag > &H)`

## 13.88.1 Macro Definition Documentation

### 13.88.1.1 \_\_FFLASFFPACK\_fger\_INL

```
#define __FFLASFFPACK_fger_INL
```

## 13.89 fflas\_fger\_mp.inl File Reference

```
#include <givaro/modular-integer.h>
#include <givaro/zring.h>
```

```
#include "fflas-ffpack/fflas/fflas_helpers.inl"
#include "fflas-ffpack/fflas/fflas_fgemm/fgemm_classical_mp.inl"
#include "fflas-ffpack/field/rns-integer.h"
#include "fflas-ffpack/field/rns-integer-mod.h"
```

## Namespaces

- namespace [FFLAS](#)

## Macros

- #define [\\_\\_FFPACK\\_fger\\_mp\\_INL](#)

## Functions

- void [fger](#) (const Givaro::Modular< Givaro::Integer > &F, const size\_t M, const size\_t N, const typename Givaro::Integer alpha, typename Givaro::Integer \*x, const size\_t incx, typename Givaro::Integer \*y, const size\_t incy, typename Givaro::Integer \*A, const size\_t lda, [MMHelper](#)< Givaro::Modular< Givaro::Integer >, [MMHelperAlgo::Classic](#), [ModeCategories::ConvertTo](#)< [ElementCategories::RNSElementTag](#) > > &H)
- template<typename [RNS](#) >  
void [fger](#) (const [FFPACK::RNSInteger](#)< [RNS](#) > &F, const size\_t M, const size\_t N, const typename [FFPACK::RNSInteger](#)< [RNS](#) >::Element alpha, typename [FFPACK::RNSInteger](#)< [RNS](#) >::Element\_ptr x, const size\_t incx, typename [FFPACK::RNSInteger](#)< [RNS](#) >::Element\_ptr y, const size\_t incy, typename [FFPACK::RNSInteger](#)< [RNS](#) >::Element\_ptr A, const size\_t lda, [MMHelper](#)< [FFPACK::RNSInteger](#)< [RNS](#) >, [MMHelperAlgo::Classic](#), [ModeCategories::DefaultTag](#) > &H)
- template<typename [RNS](#) >  
void [fger](#) (const [FFPACK::RNSIntegerMod](#)< [RNS](#) > &F, const size\_t M, const size\_t N, const type-name [FFPACK::RNSIntegerMod](#)< [RNS](#) >::Element alpha, typename [FFPACK::RNSIntegerMod](#)< [RNS](#) >::Element\_ptr x, const size\_t incx, typename [FFPACK::RNSIntegerMod](#)< [RNS](#) >::Element\_ptr y, const size\_t incy, typename [FFPACK::RNSIntegerMod](#)< [RNS](#) >::Element\_ptr A, const size\_t lda, [MMHelper](#)< [FFPACK::RNSIntegerMod](#)< [RNS](#) >, [MMHelperAlgo::Classic](#) > &H)

## 13.89.1 Macro Definition Documentation

### 13.89.1.1 [\\_\\_FFPACK\\_fger\\_mp\\_INL](#)

```
#define __FFPACK_fger_mp_INL
```

## 13.90 fflas\_freduce.h File Reference

```
#include "fflas-ffpack/fflas/fflas_simd.h"
#include "fflas-ffpack/field/field-traits.h"
#include "fflas-ffpack/utils/cast.h"
#include "fflas-ffpack/fflas/fflas_freduce.inl"
```

## Data Structures

- struct [support\\_simd\\_mod](#)< T >
- struct [support\\_fast\\_mod](#)< T >
- struct [support\\_fast\\_mod](#)< float >
- struct [support\\_fast\\_mod](#)< double >
- struct [support\\_fast\\_mod](#)< int64\_t >

## Namespaces

- namespace [FFLAS](#)

## Functions

- template<class [Field](#) >  
void [freduce](#) (const [Field](#) &F, const size\_t n, typename [Field::ConstElement\\_ptr](#) Y, const size\_t incY, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
$$\text{freduce } x \leftarrow y \bmod F.$$
- template<class [Field](#) >  
void [freduce](#) (const [Field](#) &F, const size\_t n, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
$$\text{freduce } x \leftarrow x \bmod F.$$
- template<class [Field](#) >  
void [freduce\\_constoverride](#) (const [Field](#) &F, const size\_t m, typename [Field::ConstElement\\_ptr](#) A, const size\_t incX)
- template<class [Field](#) , class ConstOtherElement\_ptr >  
void [finit](#) (const [Field](#) &F, const size\_t n, ConstOtherElement\_ptr Y, const size\_t incY, typename [Field::Element\\_ptr](#) X, const size\_t incX)
- template<class [Field](#) >  
void [finit](#) (const [Field](#) &F, const size\_t n, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
$$\text{finit Initializes } X \text{ in } F.$$
- template<class [Field](#) >  
void [freduce](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
$$\text{freduce } A \leftarrow A \bmod F.$$
- template<class [Field](#) >  
void [freduce](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) uplo, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
$$\text{freduce for square symmetric matrices}$$
- template<class [Field](#) >  
void [pfreduce](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t numths)
- template<class [Field](#) >  
void [freduce](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
$$\text{freduce } A \leftarrow B \bmod F.$$
- template<class [Field](#) >  
void [freduce\\_constoverride](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda)
- template<class [Field](#) , class OtherElement\_ptr >  
void [finit](#) (const [Field](#) &F, const size\_t m, const size\_t n, const OtherElement\_ptr B, const size\_t ldb, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
$$\text{finit } A \leftarrow B \bmod F.$$
- template<class [Field](#) >  
void [finit](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)

## 13.91 fflas\_freduce.inl File Reference

```
#include <givaro/udl.h>
#include "fflas-ffpack/fflas/fflas_fassign.h"
```

## Data Structures

- struct [HelperMod](#)< [Field](#), [ElementCategories::MachineIntTag](#) >
- struct [HelperMod](#)< [Field](#), [FFLAS::ElementCategories::MachineFloatTag](#) >
- struct [HelperMod](#)< [Field](#), [FFLAS::ElementCategories::ArbitraryPrecIntTag](#) >
- struct [HelperMod](#)< [Field](#), [FFLAS::ElementCategories::FixedPrecIntTag](#) >

## Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::vectorised](#)
- namespace [FFLAS::vectorised::unswitch](#)
- namespace [FFLAS::details](#)

## Macros

- `#define __FFLASFFPACK_fflas_freduce_INL`
- `#define FFLASFFPACK_COPY_REDUCE 32 /* TODO TO BENCHMARK LATER */`

## Functions

- `template<class T >`  
`std::enable_if<!std::is_integral< T >::value, T >::type reduce (T A, T B)`
- `template<class T >`  
`std::enable_if< std::is_integral< T >::value, T >::type reduce (T A, T B)`
- `template<> Givaro::Integer reduce (Givaro::Integer A, Givaro::Integer B)`
- `float reduce (float A, float B, float invB, float min, float max)`
- `double reduce (double A, double B, double invB, double min, double max)`
- `int64_t reduce (int64_t A, int64_t p, double invp, double min, double max, int64_t pow50rem)`
- `template<class Field >`  
`Field::Element reduce (typename Field::Element A, HelperMod< Field, ElementCategories::MachineIntTag > &H)`
- `template<class Field >`  
`Field::Element reduce (typename Field::Element A, HelperMod< Field, ElementCategories::MachineFloatTag > &H)`
- `template<class Field >`  
`Field::Element reduce (typename Field::Element A, HelperMod< Field, ElementCategories::ArbitraryPrecIntTag > &H)`
- `template<class Field >`  
`std::enable_if<!FFLAS::support_simd_mod< typenameField::Element >::value &&FFLAS::support_fast_mod< typenameField::Element >::value, void >::type modp (const Field &F, typename Field::ConstElement\_ptr U, const size_t &n, typename Field::Element\_ptr T, HelperMod< Field > &H)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type modp (const Field &F, typename Field::ConstElement\_ptr U, const size_t &n, const size_t &incX, typename Field::Element\_ptr T, HelperMod< Field > &H)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type modp (const Field &F, typename Field::ConstElement\_ptr U, const size_t &n, typename Field::Element\_ptr T)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type modp (const Field &F, typename Field::ConstElement\_ptr U, const size_t &n, const size_t &incX, typename Field::Element\_ptr T)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type freduce (const Field &F, const size_t m, typename Field::Element\_ptr A, const size_t incX, FieldCategories::ModularTag)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type freduce (const Field &F, const size_t m, typename Field::ConstElement\_ptr B, const size_t incY, typename Field::Element\_ptr A, const size_t incX, FieldCategories::ModularTag)`
- `template<class Field , class FC >`  
`void freduce (const Field &F, const size_t m, typename Field::Element\_ptr A, const size_t incX, FC)`
- `template<class Field , class FC >`  
`void freduce (const Field &F, const size_t m, typename Field::ConstElement\_ptr B, const size_t incY, type-  
name Field::Element\_ptr A, const size_t incX, FC)`

### 13.91.1 Macro Definition Documentation

#### 13.91.1.1 \_\_FFLASFFPACK\_fflas\_freduce\_INL

```
#define __FFLASFFPACK_fflas_freduce_INL
```

#### 13.91.1.2 FFLASFFPACK\_COPY\_REDUCE

```
#define FFLASFFPACK_COPY_REDUCE 32 /* TODO TO BENCHMARK LATER */
```

## 13.92 fflas\_freduce\_mp.inl File Reference

```
#include "fflas-ffpack/field/rns-integer-mod.h"
```

### Namespaces

- namespace [FFLAS](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_freduce\\_mp\\_INL](#)

### Functions

- template<> void [freduce](#) (const [FFPACK::RNSIntegerMod](#)< [FFPACK::rns\\_double](#) > &F, const size\_t n, [FFPACK::RNSIntegerMod](#)< [FFPACK::rns\\_double](#) >::Element\_ptr A, size\_t inc)
- template<> void [freduce](#) (const [FFPACK::RNSIntegerMod](#)< [FFPACK::rns\\_double](#) > &F, const size\_t m, const size\_t n, [FFPACK::rns\\_double::Element\\_ptr](#) A, size\_t lda)

### 13.92.1 Macro Definition Documentation

#### 13.92.1.1 \_\_FFLASFFPACK\_fflas\_freduce\_mp\_INL

```
#define __FFLASFFPACK_fflas_freduce_mp_INL
```

## 13.93 fflas\_freivalds.inl File Reference

### Namespaces

- namespace [FFLAS](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_freivalds\\_INL](#)

### Functions

- template<class [Field](#) >  
bool [freivalds](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t m, const size\_t n, const size\_t k, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::ConstElement\\_ptr](#) C, const size\_t ldc)

*freivalds: Freivalds **GE**neral **M**atrix **M**ultiply **R**andom **C**heck.*

### 13.93.1 Macro Definition Documentation

#### 13.93.1.1 \_\_FFLASFFPACK\_freivalds\_INL

```
#define __FFLASFFPACK_freivalds_INL
```



## 13.94 fflas\_fscal.h File Reference

```
#include "fflas_fscal.inl"
```

## 13.95 fflas\_fscal.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::vectorised](#)
- namespace [FFLAS::vectorised::unswitch](#)
- namespace [FFLAS::details](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fscal\\_INL](#)

### Functions

- [template<class Field >](#)  
std::enable\_if<![FFLAS::support\\_simd\\_mod](#)< typenameField::Element >::value &&[FFLAS::support\\_fast\\_mod](#)< typenameField::Element >::value, void >::type [scalp](#) (const [Field](#) &F, typename [Field::Element\\_ptr](#) T, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) U, const size\_t n, [HelperMod](#)< [Field](#) > &H)
- [template<class Field >](#)  
std::enable\_if< [FFLAS::support\\_fast\\_mod](#)< typenameField::Element >::value, void >::type [scalp](#) (const [Field](#) &F, typename [Field::Element\\_ptr](#) T, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) U, const size\_t n, const size\_t &incX, [HelperMod](#)< [Field](#) > &H)
- [template<class Field >](#)  
std::enable\_if< [FFLAS::support\\_fast\\_mod](#)< typenameField::Element >::value, void >::type [scalp](#) (const [Field](#) &F, typename [Field::Element\\_ptr](#) T, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) U, const size\_t n, const size\_t &incX, const size\_t &incY, [HelperMod](#)< [Field](#) > &H)
- [template<class Field >](#)  
std::enable\_if< [FFLAS::support\\_fast\\_mod](#)< typenameField::Element >::value, void >::type [scalp](#) (const [Field](#) &F, typename [Field::Element\\_ptr](#) T, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) U, const size\_t n)
- [template<class Field >](#)  
std::enable\_if< [FFLAS::support\\_fast\\_mod](#)< typenameField::Element >::value, void >::type [scalp](#) (const [Field](#) &F, typename [Field::Element\\_ptr](#) T, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) U, const size\_t n, const size\_t &incX)
- [template<class Field >](#)  
std::enable\_if< [FFLAS::support\\_fast\\_mod](#)< typenameField::Element >::value, void >::type [scalp](#) (const [Field](#) &F, typename [Field::Element\\_ptr](#) T, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) U, const size\_t n, const size\_t &incX, const size\_t &incY)
- [template<class Field >](#)  
std::enable\_if< [FFLAS::support\\_fast\\_mod](#)< typenameField::Element >::value, void >::type [fscalin](#) (const [Field](#) &F, const size\_t N, const typename [Field::Element](#) a, typename [Field::Element\\_ptr](#) X, const size\_t incX, [FieldCategories::ModularTag](#))
- [template<class Field >](#)  
std::enable\_if< [FFLAS::support\\_fast\\_mod](#)< typenameField::Element >::value, void >::type [fscal](#) (const [Field](#) &F, const size\_t N, const typename [Field::Element](#) a, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, typename [Field::Element\\_ptr](#) Y, const size\_t incY, [FieldCategories::ModularTag](#))
- [template<class Field , class FC >](#)  
void [fscalin](#) (const [Field](#) &F, const size\_t n, const typename [Field::Element](#) a, typename [Field::Element\\_ptr](#) X, const size\_t incX, FC)



- template<class [Field](#) , class FC >  
void [fscal](#) (const [Field](#) &F, const size\_t N, const typename [Field::Element](#) a, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, typename [Field::Element\\_ptr](#) Y, const size\_t incY, FC)
- template<class [Field](#) >  
void [fscal](#) (const [Field](#) &F, const size\_t n, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
$$fscal\ x \leftarrow \alpha \cdot x.$$
- template<class [Field](#) >  
void [fscal](#) (const [Field](#) &F, const size\_t n, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, typename [Field::Element\\_ptr](#) Y, const size\_t incY)  
$$fscal\ y \leftarrow \alpha \cdot x.$$
- template<> void [fscal](#) (const Givaro::DoubleDomain &, const size\_t N, const Givaro::DoubleDomain::Element a, Givaro::DoubleDomain::ConstElement\_ptr x, const size\_t incx, Givaro::DoubleDomain::Element\_ptr y, const size\_t incy)
- template<> void [fscal](#) (const Givaro::FloatDomain &, const size\_t N, const Givaro::FloatDomain::Element a, Givaro::FloatDomain::ConstElement\_ptr x, const size\_t incx, Givaro::FloatDomain::Element\_ptr y, const size\_t incy)
- template<> void [fscal](#) (const Givaro::DoubleDomain &, const size\_t N, const Givaro::DoubleDomain::Element a, Givaro::DoubleDomain::Element\_ptr y, const size\_t incy)
- template<> void [fscal](#) (const Givaro::FloatDomain &, const size\_t N, const Givaro::FloatDomain::Element a, Givaro::FloatDomain::Element\_ptr y, const size\_t incy)
- template<class [Field](#) >  
void [fscal](#) (const [Field](#) &F, const size\_t m, const size\_t n, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
$$fscal\ A \leftarrow a \cdot A.$$
- template<class [Field](#) >  
void [fscal](#) (const [Field](#) &F, const size\_t m, const size\_t n, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) B, const size\_t ldb)  
$$fscal\ B \leftarrow a \cdot A.$$

## 13.95.1 Macro Definition Documentation

### 13.95.1.1 \_\_FFLASFFPACK\_fscal\_INL

```
#define __FFLASFFPACK_fscal_INL
```

## 13.96 fflas\_fscal\_mp.inl File Reference

```
#include "fflas-ffpack/field/rns-integer.h"
#include "fflas_fscal.h"
#include "fflas_fgemm.inl"
#include "fflas-ffpack/fflas/fflas_freduce_mp.inl"
```

### Namespaces

- namespace [FFLAS](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fscal\\_mp\\_INL](#)

### Functions

- template<> void [fscal](#) (const [FFPACK::RNSInteger](#)< [FFPACK::rns\\_double](#) > &F, const size\_t n, const [FFPACK::rns\\_double::Element](#) alpha, [FFPACK::rns\\_double::Element\\_ptr](#) A, const size\_t inc)

- `template<> void fscal (const FFPACK::RNSInteger< FFPACK::rns_double > &F, const size_t n, const FFPACK::rns_double::Element alpha, FFPACK::rns_double::ConstElement_ptr A, const size_t Ainc, FFPACK::rns_double::Element_ptr B, const size_t Binc)`
- `template<> void fscaln (const FFPACK::RNSInteger< FFPACK::rns_double > &F, const size_t m, const size_t n, const FFPACK::rns_double::Element alpha, FFPACK::rns_double::Element_ptr A, const size_t lda)`
- `template<> void fscal (const FFPACK::RNSInteger< FFPACK::rns_double > &F, const size_t m, const size_t n, const FFPACK::rns_double::Element alpha, FFPACK::rns_double::ConstElement_ptr A, const size_t lda, FFPACK::rns_double::Element_ptr B, const size_t ldb)`
- `template<> void fscaln (const FFPACK::RNSIntegerMod< FFPACK::rns_double > &F, const size_t n, const typename FFPACK::RNSIntegerMod< FFPACK::rns_double >::Element alpha, typename FFPACK::RNSIntegerMod< FFPACK::rns_double >::Element_ptr A, const size_t inc)`
- `template<> void fscal (const FFPACK::RNSIntegerMod< FFPACK::rns_double > &F, const size_t n, const FFPACK::rns_double::Element alpha, FFPACK::rns_double::ConstElement_ptr A, const size_t Ainc, FFPACK::rns_double::Element_ptr B, const size_t Binc)`
- `template<> void fscaln (const FFPACK::RNSIntegerMod< FFPACK::rns_double > &F, const size_t m, const size_t n, const FFPACK::rns_double::Element alpha, FFPACK::rns_double::Element_ptr A, const size_t lda)`
- `template<> void fscal (const FFPACK::RNSIntegerMod< FFPACK::rns_double > &F, const size_t m, const size_t n, const FFPACK::rns_double::Element alpha, FFPACK::rns_double::ConstElement_ptr A, const size_t lda, FFPACK::rns_double::Element_ptr B, const size_t ldb)`

### 13.96.1 Macro Definition Documentation

#### 13.96.1.1 \_\_FFLASFFPACK\_fscal\_mp\_INL

```
#define __FFLASFFPACK_fscal_mp_INL
```

## 13.97 fflas\_fsyr2k.inl File Reference

```
#include <fflas-ffpack/utils/fflas_io.h>
```

### Namespaces

- namespace [FFLAS](#)

### Macros

- `#define __FFLASFFPACK_fflas_fsyr2k_INL`

### Functions

- `template<class Field > Field::Element_ptr fsyr2k (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc)`

*fsyr2k: Symmetric Rank 2K update*

### 13.97.1 Macro Definition Documentation

#### 13.97.1.1 \_\_FFLASFFPACK\_fflas\_fsyr2k\_INL

```
#define __FFLASFFPACK_fflas_fsyr2k_INL
```

## 13.98 fflas\_fsyrr.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::Protected](#)

### Macros

- `#define __FFLASFFPACK_fflas_fsyrr_INL`

### Functions

- `template<class NewField , class Field , class FieldMode >`  
[Field::Element\\_ptr fsyrr\\_convert](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) UpLo, const [FFLAS\\_TRANSPOSE](#) trans, const `size_t` N, const `size_t` K, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const `size_t` lda, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const `size_t` ldc, [MMHelper](#)< [Field](#), [MMHelperAlgo::Classic](#), FieldMode > &H)
- `template<class Field >`  
[Field::Element\\_ptr fsyrr](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) UpLo, const [FFLAS\\_TRANSPOSE](#) trans, const `size_t` n, const `size_t` k, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const `size_t` lda, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const `size_t` ldc)  
*fsyrr: Symmetric Rank K update*
- `template<class Field >`  
[Field::Element\\_ptr fsyrr](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) UpLo, const [FFLAS\\_TRANSPOSE](#) trans, const `size_t` N, const `size_t` K, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const `size_t` lda, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const `size_t` ldc, const [ParSeqHelper::Sequential](#) seq)
- `template<class Field >`  
[Field::Element\\_ptr fsyrr](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) UpLo, const [FFLAS\\_TRANSPOSE](#) trans, const `size_t` N, const `size_t` K, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const `size_t` lda, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const `size_t` ldc, [MMHelper](#)< [Field](#), [MMHelperAlgo::Classic](#), [ModeCategories::DefaultTag](#) > &H)
- `template<class Field >`  
[Field::Element\\_ptr fsyrr](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) UpLo, const [FFLAS\\_TRANSPOSE](#) trans, const `size_t` N, const `size_t` K, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const `size_t` lda, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const `size_t` ldc, [MMHelper](#)< [Field](#), [MMHelperAlgo::Classic](#), [ModeCategories::ConvertTo](#)< [ElementCategories::MachineFloatTag](#) >, [ParSeqHelper::Sequential](#) > &H)
- `template<class Field , class AlgoT , class ParSeqTrait >`  
[void ScalAndReduce](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) UpLo, const `size_t` N, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) A, const `size_t` lda, const [MMHelper](#)< [Field](#), AlgoT, [ModeCategories::LazyTag](#), ParSeqTrait > &H)
- `template<class Field >`  
[Field::Element\\_ptr fsyrr](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) UpLo, const [FFLAS\\_TRANSPOSE](#) trans, const `size_t` N, const `size_t` K, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const `size_t` lda, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const `size_t` ldc, [MMHelper](#)< [Field](#), [MMHelperAlgo::Classic](#), [ModeCategories::DelayedTag](#) > &H)
- `template<class Field >`  
[Field::Element\\_ptr fsyrr](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) UpLo, const [FFLAS\\_TRANSPOSE](#) trans, const `size_t` N, const `size_t` K, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const `size_t` lda, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const `size_t` ldc, [MMHelper](#)< [Field](#), [MMHelperAlgo::Classic](#), [ModeCategories::LazyTag](#) > &H)
- `template<class Field , typename Mode >`  
[Field::Element\\_ptr fsyrr](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) UpLo, const [FFLAS\\_TRANSPOSE](#) trans, const `size_t` N, const `size_t` K, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const `size_t` lda, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const `size_t` ldc, [MMHelper](#)< [Field](#), [MMHelperAlgo::DivideAndConquer](#), Mode > &H)

- `template<class Field >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const `typename Field::Element` alpha, `typename Field::ConstElement_ptr` A, const `size_t` lda, const `typename Field::Element` beta, `typename Field::Element_ptr` C, const `size_t` ldc, `MMHelper`< `Field`, `MMHelperAlgo::Classic`, `ModeCategories::DefaultBoundedTag` > &H)
- `Givaro::FloatDomain::Element_ptr fsyrk` (const `Givaro::FloatDomain` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const `Givaro::FloatDomain::Element` alpha, `Givaro::FloatDomain::ConstElement_ptr` A, const `size_t` lda, const `Givaro::FloatDomain::Element` beta, `Givaro::FloatDomain::Element_ptr` C, const `size_t` ldc, `MMHelper`< `Givaro::FloatDomain`, `MMHelperAlgo::Classic`, `ModeCategories::DefaultTag` > &H)
- `Givaro::DoubleDomain::Element_ptr fsyrk` (const `Givaro::DoubleDomain` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const `Givaro::DoubleDomain::Element` alpha, `Givaro::DoubleDomain::ConstElement_ptr` A, const `size_t` lda, const `Givaro::DoubleDomain::Element` beta, `Givaro::DoubleDomain::Element_ptr` C, const `size_t` ldc, `MMHelper`< `Givaro::DoubleDomain`, `MMHelperAlgo::Classic`, `ModeCategories::DefaultTag` > &H)
- `template<class Field >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` n, const `size_t` k, const `typename Field::Element` alpha, `typename Field::Element_ptr` A, const `size_t` lda, `typename Field::ConstElement_ptr` D, const `size_t` incD, const `typename Field::Element` beta, `typename Field::Element_ptr` C, const `size_t` ldc, const `size_t` threshold=`__FFLASFFPACK_FSYRK_THRESHOLD`)  
*fsyrk: Symmetric Rank K update with diagonal scaling*
- `template<class Field >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const `typename Field::Element` alpha, `typename Field::Element_ptr` A, const `size_t` lda, `typename Field::ConstElement_ptr` D, const `size_t` incD, const `typename Field::Element` beta, `typename Field::Element_ptr` C, const `size_t` ldc, const `ParSeqHelper::Sequential` seq, const `size_t` threshold)
- `template<class Field , class Cut , class Param >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const `typename Field::Element` alpha, `typename Field::Element_ptr` A, const `size_t` lda, `typename Field::ConstElement_ptr` D, const `size_t` incD, const `typename Field::Element` beta, `typename Field::Element_ptr` C, const `size_t` ldc, const `ParSeqHelper::Parallel`< `Cut`, `Param` > par, const `size_t` threshold)
- `template<class Field >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` n, const `size_t` k, const `typename Field::Element` alpha, `typename Field::Element_ptr` A, const `size_t` lda, `typename Field::ConstElement_ptr` D, const `size_t` incD, const `std::vector< bool >` &twoBlock, const `typename Field::Element` beta, `typename Field::Element_ptr` C, const `size_t` ldc, const `size_t` threshold=`__FFLASFFPACK_FSYRK_THRESHOLD`)  
*fsyrk: Symmetric Rank K update with diagonal scaling*

## 13.98.1 Macro Definition Documentation

### 13.98.1.1 `__FFLASFFPACK_fflas_fsyk_INL`

```
#define __FFLASFFPACK_fflas_fsyk_INL
```

## 13.99 `fflas_fsyk_strassen.inl` File Reference

```
#include <givaro/givintsqrootmod.h>
```

### Namespaces

- namespace `FFLAS`
- namespace `FFLAS::Protected`

## Macros

- `#define __FFLASFFPACK_fflas_fsyk_strassen_INL`

## Functions

- `template<class Field , class Element , class AlgoT , class ParSeqTrait >`  
`bool NeedPreScalReduction (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max,`  
`const Element &x, MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > &WH)`
- `template<class Field , class Element , class AlgoT , class ModeT , class ParSeqTrait >`  
`bool NeedPreScalReduction (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max,`  
`const Element &x, MMHelper< Field, AlgoT, ModeT, ParSeqTrait > &WH)`
- `template<class Field , class Element , class AlgoT , class ParSeqTrait >`  
`bool NeedPreAxyReduction (Element &Outmin, Element &Outmax, Element &Op1min, Element`  
`&Op1max, Element &Op2min, Element &Op2max, const Element &x, MMHelper< Field, AlgoT,`  
`ModeCategories::LazyTag, ParSeqTrait > &WH)`
- `template<class Field , class Element , class AlgoT , class ModeT , class ParSeqTrait >`  
`bool NeedPreAxyReduction (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max,`  
`Element &Op2min, Element &Op2max, const Element &x, MMHelper< Field, AlgoT, ModeT, ParSeqTrait >`  
`&WH)`
- `template<class Field , class FieldTrait >`  
`void computeS1S2 (const Field &F, const FFLAS_TRANSPOSE trans, const size_t N, const size_t K, const`  
`typename Field::Element x, const typename Field::Element y, typename Field::Element_ptr A, const size_t`  
`_t lda, typename Field::Element_ptr S, const size_t lds, typename Field::Element_ptr T, const size_t ldt,`  
`MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)`
- `template<class Field >`  
`Field::Element_ptr fsyrk (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const`  
`size_t N, const size_t K, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const`  
`size_t lda, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper<`  
`Field, MMHelperAlgo::Winograd, ModeCategories::DelayedTag, ParSeqHelper::Sequential > &H)`
- `template<class Field , class Mode >`  
`Field::Element_ptr fsyrk (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const`  
`size_t N, const size_t K, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const`  
`size_t lda, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper<`  
`Field, MMHelperAlgo::Winograd, Mode > &H)`
- `template<class Field , class FieldTrait >`  
`Field::Element_ptr fsyrk_strassen (const Field &F, const FFLAS_UPLO uplo, const FFLAS_TRANSPOSE`  
`trans, const size_t N, const size_t K, const typename Field::Element y1, const typename Field::Element`  
`y2, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, const`  
`typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field,`  
`MMHelperAlgo::Winograd, FieldTrait > &WH)`

## 13.99.1 Macro Definition Documentation

### 13.99.1.1 \_\_FFLASFFPACK\_fflas\_fsyk\_strassen\_INL

```
#define __FFLASFFPACK_fflas_fsyk_strassen_INL
```

## 13.100 fflas\_ftrmm.inl File Reference

### Namespaces

- namespace `FFLAS`

### Macros

- `#define __FFLASFFPACK_ftrmm_INL`

## Functions

- `template<class Field >`  
`void ftrmm (const Field &F, const FFLAS_SIDE Side, const FFLAS_UPLO Uplo, const FFLAS_TRANSPOSE TransA, const FFLAS_DIAG Diag, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb)`  
*ftrmm: **TR**angular **M**atrix **M**ultiply.*
- `template<class Field >`  
`void ftrmm (const Field &F, const FFLAS_SIDE Side, const FFLAS_UPLO Uplo, const FFLAS_TRANSPOSE TransA, const FFLAS_DIAG Diag, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc)`  
*ftrmm: **TR**angular **M**atrix **M**ultiply with 3 operands Computes  $C \leftarrow \alpha_{op}(A)B + \beta C$  or  $C \leftarrow \alpha B_{op}(A) + \beta C$ .*

### 13.100.1 Macro Definition Documentation

#### 13.100.1.1 \_\_FFLASFFPACK\_ftrmm\_INL

```
#define __FFLASFFPACK_ftrmm_INL
```

## 13.101 fflas\_ftrsm.inl File Reference

### Namespaces

- namespace [FFLAS](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_ftrsm\\_INL](#)

## Functions

- `template<class Field >`  
`void ftrsm (const Field &F, const FFLAS_SIDE Side, const FFLAS_UPLO Uplo, const FFLAS_TRANSPOSE TransA, const FFLAS_DIAG Diag, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb)`
- `template<class Field >`  
`void ftrsm (const Field &F, const FFLAS_SIDE Side, const FFLAS_UPLO Uplo, const FFLAS_TRANSPOSE TransA, const FFLAS_DIAG Diag, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, const ParSeqHelper::Sequential &PSH)`
- `template<class Field , class Cut , class Param >`  
`void ftrsm (const Field &F, const FFLAS_SIDE Side, const FFLAS_UPLO Uplo, const FFLAS_TRANSPOSE TransA, const FFLAS_DIAG Diag, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, const ParSeqHelper::Parallel< Cut, Param > &PSH)`
- `template<class Field , class ParSeqTrait = ParSeqHelper::Sequential>`  
`void ftrsm (const Field &F, const FFLAS_SIDE Side, const FFLAS_UPLO Uplo, const FFLAS_TRANSPOSE TransA, const FFLAS_DIAG Diag, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, TRSMHelper< StructureHelper::Recursive, ParSeqTrait > &H)`

### 13.101.1 Macro Definition Documentation

#### 13.101.1.1 \_\_FFLASFFPACK\_ftrsm\_INL

```
#define __FFLASFFPACK_ftrsm_INL
```

## 13.102 fflas\_ftrsm\_mp.inl File Reference

triangular system with matrix right hand side over multiprecision domain (either over  $\mathbb{Z}$  or over  $\mathbb{Z}/p\mathbb{Z}$ )

```
#include <cmath>
#include <givaro/modular-integer.h>
#include <givaro/givinteger.h>
#include "fflas-ffpack/fflas/fflas_bounds.inl"
#include "fflas-ffpack/fflas/fflas_level3.inl"
#include "fflas-ffpack/field/rns-integer-mod.h"
#include "fflas-ffpack/field/rns-integer.h"
```

### Namespaces

- namespace [FFLAS](#)

### Macros

- #define [\\_\\_FFPACK\\_ftrsm\\_mp\\_INL](#)

### Functions

- void [ftrsm](#) (const Givaro::Modular< Givaro::Integer > &F, const [FFLAS\\_SIDE](#) Side, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, const Givaro::Integer alpha, const Givaro::Integer \*A, const size\_t lda, Givaro::Integer \*B, const size\_t ldb)
- void [cblas\\_impftrsm](#) (const enum [FFLAS\\_ORDER](#) Order, const enum [FFLAS\\_SIDE](#) Side, const enum [FFLAS\\_UPLO](#) Uplo, const enum [FFLAS\\_TRANSPOSE](#) TransA, const enum [FFLAS\\_DIAG](#) Diag, const int M, const int N, const [FFPACK::rns\\_double\\_elt](#) alpha, [FFPACK::rns\\_double\\_elt\\_cstptr](#) A, const int lda, [FFPACK::rns\\_double\\_elt\\_ptr](#) B, const int ldb)

### 13.102.1 Detailed Description

triangular system with matrix right hand side over multiprecision domain (either over  $\mathbb{Z}$  or over  $\mathbb{Z}/p\mathbb{Z}$ )

### 13.102.2 Macro Definition Documentation

#### 13.102.2.1 \_\_FFPACK\_ftrsm\_mp\_INL

```
#define __FFPACK_ftrsm_mp_INL
```

## 13.103 fflas\_ftrsv.inl File Reference

### Namespaces

- namespace [FFLAS](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_ftrsv\\_INL](#)

### Functions

- template<class [Field](#) >  
void [ftrsv](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) X, int incX)

*ftrsv: TRIangular System solve with Vector Computes  $X \leftarrow \text{op}(A^{-1})X$*

### 13.103.1 Macro Definition Documentation

#### 13.103.1.1 \_\_FFLASFFPACK\_ftrsv\_INL

```
#define __FFLASFFPACK_ftrsv_INL
```

### 13.104 fflas\_helpers.inl File Reference

```
#include "fflas-ffpack/field/field-traits.h"
#include "fflas-ffpack/paladin/parallel.h"
#include "fflas-ffpack/utils/flimits.h"
#include <algorithm>
```

#### Data Structures

- struct [Auto](#)
- struct [Classic](#)
- struct [DivideAndConquer](#)
- struct [Winograd](#)
- struct [WinogradPar](#)
- struct [Bini](#)
- struct [AlgoChooser](#)< [ModeT](#), [ParSeq](#) >
- struct [AlgoChooser](#)< [ModeCategories::ConvertTo](#)< [ElementCategories::RNSElementTag](#) >, [ParSeq](#) >
- struct [MMHelper](#)< [Field](#), [AlgoTrait](#), [ModeCategories::DefaultTag](#), [ParSeqTrait](#) >

*FGEMM Helper for Default and ConvertTo modes of operation.*

- struct [MMHelper](#)< [Field](#), [AlgoTrait](#), [ModeCategories::ConvertTo](#)< [Dest](#) >, [ParSeqTrait](#) >
- struct [MMHelper](#)< [Field](#), [AlgoTrait](#), [ModeTrait](#), [ParSeqTrait](#) >
- struct [Recursive](#)
- struct [Iterative](#)
- struct [Hybrid](#)
- struct [TRSMHelper](#)< [ReclterTrait](#), [ParSeqTrait](#) >

*TRSM Helper.*

#### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::Protected](#)
- namespace [FFLAS::MMHelperAlgo](#)
- namespace [FFLAS::StructureHelper](#)

*StructureHelper for ftrsm.*

#### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_fflas\\_mmhelper\\_INL](#)

#### Functions

- template<class [Field](#) >  
int [WinogradSteps](#) (const [Field](#) &F, const size\_t &m)  
*Computes the number of recursive levels to perform.*
- template<class DFE >  
size\_t [min\\_types](#) (const DFE &k)
- template<> size\_t [min\\_types](#) (const [Reclnt::rint](#)< 6 > &k)
- template<> size\_t [min\\_types](#) (const [Reclnt::rint](#)< 7 > &k)
- template<> size\_t [min\\_types](#) (const [Reclnt::rint](#)< 8 > &k)



- `template<> size_t min_types (const Reclnt::rint< 9 > &k)`
- `template<> size_t min_types (const Reclnt::rint< 10 > &k)`
- `template<> size_t min_types (const Givaro::Integer &k)`
- `template<class T >`  
    `bool unfit (T x)`
- `template<> bool unfit (int64_t x)`
- `template<size_t K>`  
    `bool unfit (Reclnt::rint< K > x)`
- `template<> bool unfit (Reclnt::rint< 6 > x)`

### 13.104.1 Macro Definition Documentation

#### 13.104.1.1 \_\_FFLASFFPACK\_fflas\_fflas\_mmhelper\_INL

```
#define __FFLASFFPACK_fflas_fflas_mmhelper_INL
```

## 13.105 igemm.doxy File Reference

## 13.106 igemm.h File Reference

```
#include "igemm_kernels.h"
#include "igemm_tools.h"
#include "fflas-ffpack/utils/fflas_memory.h"
#include "igemm.inl"
```

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::Protected](#)

### Enumerations

- enum [number\\_kind](#) { [zero](#) =0 , [one](#) =1 , [mone](#) =-1 , [other](#) =2 }

### Functions

- `template<enum FFLAS\_TRANSPOSE tA, enum FFLAS\_TRANSPOSE tB>`  
    `void igemm\_colmajor (size_t rows, size_t cols, size_t depth, const int64_t alpha, const int64_t *A, size_t lda, const int64_t *B, size_t ldb, int64_t *C, size_t ldc)`
- `template<enum FFLAS\_TRANSPOSE tA, enum FFLAS\_TRANSPOSE tB, enum number\_kind alpha_kind>`  
    `void igemm\_colmajor (size_t rows, size_t cols, size_t depth, const int64_t alpha, const int64_t *A, size_t lda, const int64_t *B, size_t ldb, int64_t *C, size_t ldc)`
- `void igemm (const enum FFLAS\_TRANSPOSE TransA, const enum FFLAS\_TRANSPOSE TransB, size_t rows, size_t cols, size_t depth, const int64_t alpha, const int64_t *A, size_t lda, const int64_t *B, size_t ldb, const int64_t beta, int64_t *C, size_t ldc)`
- `void igemm\_ (const enum FFLAS\_ORDER Order, const enum FFLAS\_TRANSPOSE TransA, const enum FFLAS\_TRANSPOSE TransB, const size_t M, const size_t N, const size_t K, const int64_t alpha, const int64_t *A, const size_t lda, const int64_t *B, const size_t ldb, const int64_t beta, int64_t *C, const size_t ldc)`

## 13.107 igemm.inl File Reference

```
#include "fflas-ffpack/utils/fflas_memory.h"
```

## Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::Protected](#)

## Macros

- `#define` [\\_\\_FFLASFFPACK\\_fflas\\_igemm\\_igemm\\_INL](#)

## Functions

- `template<enum FFLAS\_TRANSPOSE tA, enum FFLAS\_TRANSPOSE tB>`  
void [igemm\\_colmajor](#) (size\_t rows, size\_t cols, size\_t depth, const int64\_t alpha, const int64\_t \*A, size\_t lda, const int64\_t \*B, size\_t ldb, int64\_t \*C, size\_t ldc)
- `template<enum FFLAS\_TRANSPOSE tA, enum FFLAS\_TRANSPOSE tB, enum number\_kind alpha_kind>`  
void [igemm\\_colmajor](#) (size\_t rows, size\_t cols, size\_t depth, const int64\_t alpha, const int64\_t \*A, size\_t lda, const int64\_t \*B, size\_t ldb, int64\_t \*C, size\_t ldc)
- void [igemm](#) (const enum [FFLAS\\_TRANSPOSE](#) TransA, const enum [FFLAS\\_TRANSPOSE](#) TransB, size\_t rows, size\_t cols, size\_t depth, const int64\_t alpha, const int64\_t \*A, size\_t lda, const int64\_t \*B, size\_t ldb, const int64\_t beta, int64\_t \*C, size\_t ldc)
- void [igemm\\_](#) (const enum [FFLAS\\_ORDER](#) Order, const enum [FFLAS\\_TRANSPOSE](#) TransA, const enum [FFLAS\\_TRANSPOSE](#) TransB, const size\_t M, const size\_t N, const size\_t K, const int64\_t alpha, const int64\_t \*A, const size\_t lda, const int64\_t \*B, const size\_t ldb, const int64\_t beta, int64\_t \*C, const size\_t ldc)

## 13.107.1 Macro Definition Documentation

### 13.107.1.1 [\\_\\_FFLASFFPACK\\_fflas\\_igemm\\_igemm\\_INL](#)

```
#define __FFLASFFPACK_fflas_igemm_igemm_INL
```

## 13.108 [igemm\\_kernels.h](#) File Reference

```
#include "igemm_kernels.inl"
```

## Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::details](#)

## Functions

- `template<enum number\_kind K>`  
void [igebb44](#) (size\_t i, size\_t j, size\_t depth, size\_t pdeth, const int64\_t alpha, const int64\_t \*bIA, const int64\_t \*bIB, int64\_t \*C, size\_t ldc)
- `template<enum number\_kind K>`  
void [igebb24](#) (size\_t i, size\_t j, size\_t depth, size\_t pdeth, const int64\_t alpha, const int64\_t \*bIA, const int64\_t \*bIB, int64\_t \*C, size\_t ldc)
- `template<enum number\_kind K>`  
void [igebb14](#) (size\_t i, size\_t j, size\_t depth, size\_t pdeth, const int64\_t alpha, const int64\_t \*bIA, const int64\_t \*bIB, int64\_t \*C, size\_t ldc)
- `template<enum number\_kind K>`  
void [igebb41](#) (size\_t i, size\_t j, size\_t depth, size\_t pdeth, const int64\_t alpha, const int64\_t \*bIA, const int64\_t \*bIB, int64\_t \*C, size\_t ldc)
- `template<enum number\_kind K>`  
void [igebb21](#) (size\_t i, size\_t j, size\_t depth, size\_t pdeth, const int64\_t alpha, const int64\_t \*bIA, const int64\_t \*bIB, int64\_t \*C, size\_t ldc)

- template<enum [number\\_kind](#) K>  
void [igebb11](#) (size\_t i, size\_t j, size\_t depth, size\_t pdeth, const int64\_t alpha, const int64\_t \*bIA, const int64\_t \*bIB, int64\_t \*C, size\_t ldc)
- template<enum [number\\_kind](#) K>  
void [igebp](#) (size\_t rows, size\_t cols, size\_t depth, const int64\_t alpha, const int64\_t \*blockA, size\_t lda, const int64\_t \*blockB, size\_t ldb, int64\_t \*C, size\_t ldc)

## 13.109 igemm\_kernels.inl File Reference

```
#include "fflas-ffpack/utils/fflas_memory.h"
#include "igemm_tools.h"
```

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::details](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_igemm\\_igemm\\_kernels\\_INL](#)

### Functions

- template<enum [number\\_kind](#) K>  
void [igebb44](#) (size\_t i, size\_t j, size\_t depth, size\_t pdeth, const int64\_t alpha, const int64\_t \*bIA, const int64\_t \*bIB, int64\_t \*C, size\_t ldc)
- template<enum [number\\_kind](#) K>  
void [igebb24](#) (size\_t i, size\_t j, size\_t depth, size\_t pdeth, const int64\_t alpha, const int64\_t \*bIA, const int64\_t \*bIB, int64\_t \*C, size\_t ldc)
- template<enum [number\\_kind](#) K>  
void [igebb14](#) (size\_t i, size\_t j, size\_t depth, size\_t pdeth, const int64\_t alpha, const int64\_t \*bIA, const int64\_t \*bIB, int64\_t \*C, size\_t ldc)
- template<enum [number\\_kind](#) K>  
void [igebb41](#) (size\_t i, size\_t j, size\_t depth, size\_t pdeth, const int64\_t alpha, const int64\_t \*bIA, const int64\_t \*bIB, int64\_t \*C, size\_t ldc)
- template<enum [number\\_kind](#) K>  
void [igebb21](#) (size\_t i, size\_t j, size\_t depth, size\_t pdeth, const int64\_t alpha, const int64\_t \*bIA, const int64\_t \*bIB, int64\_t \*C, size\_t ldc)
- template<enum [number\\_kind](#) K>  
void [igebb11](#) (size\_t i, size\_t j, size\_t depth, size\_t pdeth, const int64\_t alpha, const int64\_t \*bIA, const int64\_t \*bIB, int64\_t \*C, size\_t ldc)
- template<enum [number\\_kind](#) K>  
void [igebp](#) (size\_t rows, size\_t cols, size\_t depth, const int64\_t alpha, const int64\_t \*blockA, size\_t lda, const int64\_t \*blockB, size\_t ldb, int64\_t \*C, size\_t ldc)

### 13.109.1 Macro Definition Documentation

#### 13.109.1.1 [\\_\\_FFLASFFPACK\\_fflas\\_igemm\\_igemm\\_kernels\\_INL](#)

```
#define __FFLASFFPACK_fflas_igemm_igemm_kernels_INL
```

## 13.110 igemm\_tools.h File Reference

```
#include "igemm_tools.inl"
```

## Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::details](#)

## Functions

- `template<size_t k, bool transpose>`  
void [pack\\_lhs](#) (int64\_t \*XX, const int64\_t \*X, size\_t ldx, size\_t rows, size\_t cols)
- `template<size_t k, bool transpose>`  
void [pack\\_rhs](#) (int64\_t \*XX, const int64\_t \*X, size\_t ldx, size\_t rows, size\_t cols)
- void [gebp](#) (size\_t rows, size\_t cols, size\_t depth, int64\_t \*C, size\_t ldc, const int64\_t \*blockA, size\_t lda, const int64\_t \*BlockB, size\_t ldb, int64\_t \*BlockW)
- void [BlockingFactor](#) (size\_t &m, size\_t &n, size\_t &k)

## 13.111 igemm\_tools.inl File Reference

```
#include "fflas-ffpack/fflas/fflas_simd.h"
```

## Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::details](#)

## Macros

- `#define` [\\_\\_FFLASFFPACK\\_fflas\\_igemm\\_igemm\\_tools\\_INL](#)

## Functions

- `template<size_t k, bool transpose>`  
void [pack\\_rhs](#) (int64\_t \*XX, const int64\_t \*X, size\_t ldx, size\_t rows, size\_t cols)
- `template<size_t k, bool transpose>`  
void [pack\\_lhs](#) (int64\_t \*XX, const int64\_t \*X, size\_t ldx, size\_t rows, size\_t cols)
- void [BlockingFactor](#) (size\_t &m, size\_t &n, size\_t &k)

### 13.111.1 Macro Definition Documentation

#### 13.111.1.1 \_\_FFLASFFPACK\_fflas\_igemm\_igemm\_tools\_INL

```
#define __FFLASFFPACK_fflas_igemm_igemm_tools_INL
```

## 13.112 fflas\_level1.inl File Reference

## Namespaces

- namespace [FFLAS](#)

## Macros

- `#define` [\\_\\_FFLASFFPACK\\_fflas\\_fflas\\_level1\\_INL](#)

## Functions

- template<class [Field](#) >  
void [freduce](#) (const [Field](#) &F, const size\_t n, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
 $freduce\ x \leftarrow x \bmod F.$
- template<class [Field](#) >  
void [freduce](#) (const [Field](#) &F, const size\_t n, typename [Field::ConstElement\\_ptr](#) Y, const size\_t incY, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
 $freduce\ x \leftarrow y \bmod F.$
- template<class [Field](#), class OtherElement\_ptr >  
void [finit](#) (const [Field](#) &F, const size\_t n, const OtherElement\_ptr Y, const size\_t incY, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
 $finit\ x \leftarrow y \bmod F.$
- template<class [Field](#) >  
void [finit](#) (const [Field](#) &F, const size\_t n, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
 $finit$  Initializes  $X$  in  $F$ .
- template<class [Field](#), class OtherElement\_ptr >  
void [fconvert](#) (const [Field](#) &F, const size\_t n, OtherElement\_ptr X, const size\_t incX, typename [Field::ConstElement\\_ptr](#) Y, const size\_t incY)  
 $fconvert\ x \leftarrow y \bmod F.$
- template<class [Field](#) >  
void [fnegin](#) (const [Field](#) &F, const size\_t n, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
 $fnegin\ x \leftarrow -x.$
- template<class [Field](#) >  
void [fneg](#) (const [Field](#) &F, const size\_t n, typename [Field::ConstElement\\_ptr](#) Y, const size\_t incY, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
 $fneg\ x \leftarrow -y.$
- template<class [Field](#) >  
void [fzero](#) (const [Field](#) &F, const size\_t n, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
 $fzero : A \leftarrow 0.$
- template<class [Field](#), class RandIter >  
void [frand](#) (const [Field](#) &F, RandIter &G, const size\_t n, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
 $frand : A \leftarrow random.$
- template<class [Field](#) >  
bool [fiszero](#) (const [Field](#) &F, const size\_t n, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX)  
 $fiszero : test\ X = 0.$
- template<class [Field](#) >  
bool [fequal](#) (const [Field](#) &F, const size\_t n, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, typename [Field::ConstElement\\_ptr](#) Y, const size\_t incY)  
 $fequal : test\ X = Y.$
- template<class [Field](#) >  
void [fassign](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) Y, const size\_t incY, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
 $fassign : x \leftarrow y.$
- template<class [Field](#) >  
void [fscaln](#) (const [Field](#) &F, const size\_t n, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
 $fscaln\ x \leftarrow \alpha \cdot x.$
- template<class [Field](#) >  
void [fscal](#) (const [Field](#) &F, const size\_t n, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, typename [Field::Element\\_ptr](#) Y, const size\_t incY)  
 $fscal\ y \leftarrow \alpha \cdot x.$
- template<class [Field](#) >  
void [faxpy](#) (const [Field](#) &F, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, typename [Field::Element\\_ptr](#) Y, const size\_t incY)

- $$\text{faxpy} : y \leftarrow \alpha \cdot x + y.$$
  - template<class `Field` >  
void `faxpby` (const `Field` &F, const `size_t` N, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` X, const `size_t` incX, const typename `Field::Element` beta, typename `Field::Element_ptr` Y, const `size_t` incY)  

$$\text{faxpby} : y \leftarrow \alpha \cdot x + \beta \cdot y.$$
- template<class `Field` >  
`Field::Element` `fdot` (const `Field` &F, const `size_t` N, typename `Field::ConstElement_ptr` X, const `size_t` incX, typename `Field::ConstElement_ptr` Y, const `size_t` incY)  

$$\text{fdot} : \text{dot product } x^T y.$$
- template<class `Field` >  
`Field::Element` `fdot` (const `Field` &F, const `size_t` N, typename `Field::ConstElement_ptr` x, const `size_t` incx, typename `Field::ConstElement_ptr` y, const `size_t` incy, const `ParSeqHelper::Sequential` seq)
- template<typename `Field` , class `Cut` , class `Param` >  
`Field::Element` `fdot` (const `Field` &F, const `size_t` N, typename `Field::ConstElement_ptr` X, const `size_t` incX, typename `Field::ConstElement_ptr` Y, const `size_t` incY, const `ParSeqHelper::Parallel`< `Cut`, `Param` > par)
- template<class `Field` >  
void `fswap` (const `Field` &F, const `size_t` N, typename `Field::Element_ptr` X, const `size_t` incX, typename `Field::Element_ptr` Y, const `size_t` incY)  

$$\text{fswap} : X \leftrightarrow Y.$$
- template<class `Field` >  
void `pfadd` (const `Field` &F, const `size_t` M, const `size_t` N, typename `Field::ConstElement_ptr` A, const `size_t` lda, typename `Field::ConstElement_ptr` B, const `size_t` ldb, typename `Field::Element_ptr` C, const `size_t` ldc, const `size_t` numths)
- template<class `Field` >  
void `pfsub` (const `Field` &F, const `size_t` M, const `size_t` N, typename `Field::ConstElement_ptr` A, const `size_t` lda, typename `Field::ConstElement_ptr` B, const `size_t` ldb, typename `Field::Element_ptr` C, const `size_t` ldc, const `size_t` numths)
- template<class `Field` >  
void `pfaddin` (const `Field` &F, const `size_t` M, const `size_t` N, typename `Field::ConstElement_ptr` B, const `size_t` ldb, typename `Field::Element_ptr` C, const `size_t` ldc, `size_t` numths)
- template<class `Field` >  
void `pfsubin` (const `Field` &F, const `size_t` M, const `size_t` N, typename `Field::ConstElement_ptr` B, const `size_t` ldb, typename `Field::Element_ptr` C, const `size_t` ldc, `size_t` numths)
- template<class `Field` >  
void `fadd` (const `Field` &F, const `size_t` N, typename `Field::ConstElement_ptr` A, const `size_t` inca, typename `Field::ConstElement_ptr` B, const `size_t` incb, typename `Field::Element_ptr` C, const `size_t` incc)
- template<class `Field` >  
void `fsub` (const `Field` &F, const `size_t` N, typename `Field::ConstElement_ptr` A, const `size_t` inca, typename `Field::ConstElement_ptr` B, const `size_t` incb, typename `Field::Element_ptr` C, const `size_t` incc)
- template<class `Field` >  
void `faddin` (const `Field` &F, const `size_t` N, typename `Field::ConstElement_ptr` B, const `size_t` incb, typename `Field::Element_ptr` C, const `size_t` incc)
- template<class `Field` >  
void `fsubin` (const `Field` &F, const `size_t` N, typename `Field::ConstElement_ptr` B, const `size_t` incb, typename `Field::Element_ptr` C, const `size_t` incc)
- template<class `Field` >  
void `fadd` (const `Field` &F, const `size_t` N, typename `Field::ConstElement_ptr` A, const `size_t` inca, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` B, const `size_t` incb, typename `Field::Element_ptr` C, const `size_t` incc)

### 13.112.1 Macro Definition Documentation

#### 13.112.1.1 `__FFLASFFPACK_fflas_fflas_level1_INL`

```
#define __FFLASFFPACK_fflas_fflas_level1_INL
```

## 13.113 fflas\_level2.inl File Reference

```
#include "givaro/zring.h"
```

### Namespaces

- namespace [FFLAS](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_fflas\\_level2\\_INL](#)

### Functions

- template<class [Field](#) >  
void [fassign](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*fassign :  $A \leftarrow B$ .*
- template<class [Field](#) >  
void [fzero](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*fzero :  $A \leftarrow 0$ .*
- template<class [Field](#) >  
void [fzero](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) shape, const [FFLAS\\_DIAG](#) diag, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*fzero :  $A \leftarrow 0$  for a triangular matrix.*
- template<class [Field](#), class Randlter >  
void [frand](#) (const [Field](#) &F, Randlter &G, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*frand :  $A \leftarrow \text{random}$ .*
- template<class [Field](#) >  
bool [fequal](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb)  
*fequal : test  $A = B$ .*
- template<class [Field](#) >  
bool [fiszero](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda)  
*fiszero : test  $A = 0$ .*
- template<class [Field](#) >  
void [fidentity](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda, const typename [Field::Element](#) &d)  
*creates a diagonal matrix*
- template<class [Field](#) >  
void [fidentity](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*creates a diagonal matrix*
- template<class [Field](#) >  
void [freduce](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*freduce  $A \leftarrow A \bmod F$ .*
- template<class [Field](#) >  
void [freduce](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) uplo, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*freduce for square symmetric matrices*
- template<class [Field](#) >  
void [freduce](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*freduce  $A \leftarrow B \bmod F$ .*

- `template<class Field , class OtherElement_ptr >`  
`void finit (const Field &F, const size_t m, const size_t n, const OtherElement_ptr B, const size_t ldb, typename Field::Element_ptr A, const size_t lda)`  

$$\text{finit } A \leftarrow B \bmod F.$$
- `template<class Field , class OtherElement_ptr >`  
`void finit (const Field &F, const size_t m, const size_t n, typename Field::Element_ptr A, const size_t lda)`  

$$\text{finit initializes } A \text{ in } F\mathbb{Z}.$$
- `template<class Field , class OtherElement_ptr >`  
`void fconvert (const Field &F, const size_t m, const size_t n, OtherElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb)`  

$$\text{fconvert } A \leftarrow B \bmod F.$$
- `template<class Field >`  
`void fnegin (const Field &F, const size_t m, const size_t n, typename Field::Element_ptr A, const size_t lda)`  

$$\text{fnegin } A \leftarrow -A.$$
- `template<class Field >`  
`void fneg (const Field &F, const size_t m, const size_t n, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr A, const size_t lda)`  

$$\text{fneg } A \leftarrow -B.$$
- `template<class Field >`  
`void fscaln (const Field &F, const size_t m, const size_t n, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda)`  

$$\text{fscaln } A \leftarrow a \cdot A.$$
- `template<class Field >`  
`void fscal (const Field &F, const size_t m, const size_t n, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb)`  

$$\text{fscal } B \leftarrow a \cdot A.$$
- `template<class Field >`  
`void faxpy (const Field &F, const size_t m, const size_t n, const typename Field::Element alpha, typename Field::ConstElement_ptr X, const size_t ldx, typename Field::Element_ptr Y, const size_t ldy)`  

$$\text{faxpy : } y \leftarrow \alpha \cdot x + y.$$
- `template<class Field >`  
`void faxpby (const Field &F, const size_t m, const size_t n, const typename Field::Element alpha, typename Field::ConstElement_ptr X, const size_t ldx, const typename Field::Element beta, typename Field::Element_ptr Y, const size_t ldy)`  

$$\text{faxpby : } y \leftarrow \alpha \cdot x + \beta \cdot y.$$
- `template<class Field >`  
`void fmove (const Field &F, const size_t m, const size_t n, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb)`  

$$\text{fmove : } A \leftarrow B \text{ and } B \leftarrow 0.$$
- `template<class Field >`  
`void fadd (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc)`  

$$\text{fadd : matrix addition.}$$
- `template<class Field >`  
`void fsub (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc)`  

$$\text{fsub : matrix subtraction.}$$
- `template<class Field >`  
`void fsubin (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc)`  

$$\text{fsubin } C = C - B$$
- `template<class Field >`  
`void fadd (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element alpha, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc)`



- fadd : matrix addition with scaling.*
- template<class [Field](#) >  
void [faddin](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) C, const size\_t ldc)  
*faddin*
  - template<class [Field](#) >  
void [faddin](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) uplo, const size\_t N, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) C, const size\_t ldc)  
*fadding for symmetric matrices*
  - template<class [Field](#) >  
[Field::Element\\_ptr](#) [fgemv](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) TransA, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) Y, const size\_t incY)  
*finite prime Field GEneral Matrix Vector multiplication.*
  - template<class [Field](#) >  
void [fger](#) (const [Field](#) &F, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) x, const size\_t incx, typename [Field::ConstElement\\_ptr](#) y, const size\_t incy, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*fger: rank one update of a general matrix*
  - template<class [Field](#) >  
void [ftrsv](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) X, int incX)  
*ftrsv: TRIangular System solve with Vector Computes  $X \leftarrow \text{op}(A^{-1})X$*
  - template<class [Field](#) >  
size\_t [bitsize](#) (const [Field](#) &F, size\_t M, size\_t N, const typename [Field::ConstElement\\_ptr](#) A, size\_t lda)  
*bitsize: Computes the largest bitsize of the matrix' coefficients.*
  - template<> size\_t [bitsize](#)< [Givaro::ZRing](#)< [Givaro::Integer](#) > > (const [Givaro::ZRing](#)< [Givaro::Integer](#) > &F, size\_t M, size\_t N, const [Givaro::Integer](#) \*A, size\_t lda)
  - template<class [Field](#) >  
void [ftrmv](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) X, int incX)  
*ftrsm: TRIangular Matrix Vector prodcut Computes  $X \leftarrow \text{op}(A)X$*

### 13.113.1 Macro Definition Documentation

#### 13.113.1.1 \_\_FFLASFFPACK\_fflas\_fflas\_level2\_INL

```
#define __FFLASFFPACK_fflas_fflas_level2_INL
```

## 13.114 fflas\_level3.inl File Reference

```
#include "fflas_bounds.inl"
#include "fflas_helpers.inl"
#include "fflas-ffpack/paladin/parallel.h"
```

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::Protected](#)

## Macros

- `#define __FFLASFFPACK_fflas_fflas_level3_INL`
- `#define __FFLAS__TRSM_READONLY`

## Functions

- `template<class Field >`  
`void MatF2MatD_Triangular (const Field &F, Givaro::DoubleDomain::Element_ptr S, const size_t lds, type-`  
`name Field::ConstElement_ptr const E, const size_t lde, const size_t m, const size_t n)`
- `template<class Field >`  
`void MatF2MatFI_Triangular (const Field &F, Givaro::FloatDomain::Element_ptr S, const size_t lds, typename`  
`Field::ConstElement_ptr const E, const size_t lde, const size_t m, const size_t n)`
- `template<class Field >`  
`void ftrsm (const Field &F, const FFLAS_SIDE Side, const FFLAS_UPLO Uplo, const FFLAS_TRANSPOSE`  
`TransA, const FFLAS_DIAG Diag, const size_t M, const size_t N, const typename Field::Element alpha,`  
`typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb)`  
*ftrsm: **TR**angular **S**ystem solve with **M**atrix.*
- `template<class Field >`  
`void ftrmm (const Field &F, const FFLAS_SIDE Side, const FFLAS_UPLO Uplo, const FFLAS_TRANSPOSE`  
`TransA, const FFLAS_DIAG Diag, const size_t M, const size_t N, const typename Field::Element alpha,`  
`typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb)`  
*ftrmm: **TR**angular **M**atrix **M**ultiply.*
- `template<class Field >`  
`void ftrmm (const Field &F, const FFLAS_SIDE Side, const FFLAS_UPLO Uplo, const FFLAS_TRANSPOSE`  
`TransA, const FFLAS_DIAG Diag, const size_t M, const size_t N, const typename Field::Element alpha,`  
`typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t`  
`ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc)`  
*ftrmm: **TR**angular **M**atrix **M**ultiply with 3 operands Computes  $C \leftarrow \alpha \text{op}(A)B + \text{beta}C$  or  $C \leftarrow \alpha B \text{op}(A) + \text{beta}C$ .*
- `template<class Field >`  
`Field::Element_ptr fsyrk (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const`  
`size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const`  
`size_t lda, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc)`  
*fsyrk: Symmetric Rank K update*
- `template<class Field , typename FieldTrait >`  
`Field::Element_ptr fsyrk_strassen (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE`  
`trans, const size_t N, const size_t K, const typename Field::Element y1, const typename Field::Element`  
`y2, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, const`  
`typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field,`  
`MMHelperAlgo::Winograd, FieldTrait > &H)`
- `template<class Field >`  
`Field::Element_ptr fsyr2k (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans,`  
`const size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement_ptr A,`  
`const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element beta,`  
`typename Field::Element_ptr C, const size_t ldc)`  
*fsyr2k: Symmetric Rank 2K update*
- `template<class Field >`  
`Field::Element_ptr fsyrk (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const`  
`size_t n, const size_t k, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t`  
`lda, typename Field::ConstElement_ptr D, const size_t incD, const typename Field::Element beta, typename`  
`Field::Element_ptr C, const size_t ldc, const size_t threshold=__FFLASFFPACK_FSYRK_THRESHOLD)`  
*fsyrk: Symmetric Rank K update with diagonal scaling*
- `template<class Field >`  
`Field::Element_ptr fsyrk (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const`  
`size_t N, const size_t K, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t`  
`lda, typename Field::ConstElement_ptr D, const size_t incD, const typename Field::Element beta, typename`  
`Field::Element_ptr C, const size_t ldc, const ParSeqHelper::Sequential seq, const size_t threshold)`

- template<class [Field](#) , class [Cut](#) , class [Param](#) >  
[Field::Element\\_ptr](#) fsyrk (const [Field](#) &F, const [FFLAS\\_UPLO](#) UpLo, const [FFLAS\\_TRANSPOSE](#) trans, const size\_t N, const size\_t K, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) D, const size\_t incD, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, const [ParSeqHelper::Parallel](#)< [Cut](#), [Param](#) > par, const size\_t threshold)  
*fsyrk: Symmetric Rank K update with diagonal scaling*
- template<class [Field](#) >  
[Field::Element\\_ptr](#) fsyrk (const [Field](#) &F, const [FFLAS\\_UPLO](#) UpLo, const [FFLAS\\_TRANSPOSE](#) trans, const size\_t n, const size\_t k, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) D, const size\_t incD, const std::vector< bool > &two←Block, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, const size\_t threshold=\_\_FFLASFFPACK\_FSYRK\_THRESHOLD)  
*fsyrk: Symmetric Rank K update with diagonal scaling*
- template<typename [Field](#) >  
[Field::Element\\_ptr](#) fgemm (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t m, const size\_t n, const size\_t k, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc)  
*fgemm: Field GENERAL Matrix Multiply.*
- template<typename [Field](#) >  
[Field::Element\\_ptr](#) fgemm (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t m, const size\_t n, const size\_t k, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, const [ParSeqHelper::Sequential](#) seq)
- template<typename [Field](#) , class [Cut](#) , class [Param](#) >  
[Field::Element\\_ptr](#) fgemm (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t m, const size\_t n, const size\_t k, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, const [ParSeqHelper::Parallel](#)< [Cut](#), [Param](#) > par)
- template<typename [Field](#) >  
[Field::Element\\_ptr](#) pfgemm (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t m, const size\_t n, const size\_t k, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, size\_t numthreads=0)
- template<class [Field](#) >  
[Field::Element](#) \* pfgemm\_1D\_rec (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t m, const size\_t n, const size\_t k, const typename [Field::Element](#) alpha, const typename [Field::Element\\_ptr](#) A, const size\_t lda, const typename [Field::Element\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element](#) \*C, const size\_t ldc, size\_t seuil)
- template<class [Field](#) >  
[Field::Element](#) \* pfgemm\_2D\_rec (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t m, const size\_t n, const size\_t k, const typename [Field::Element](#) alpha, const typename [Field::Element\\_ptr](#) A, const size\_t lda, const typename [Field::Element\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element](#) \*C, const size\_t ldc, size\_t seuil)
- template<class [Field](#) >  
[Field::Element](#) \* pfgemm\_3D\_rec (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t m, const size\_t n, const size\_t k, const typename [Field::Element](#) alpha, const typename [Field::Element\\_ptr](#) A, const size\_t lda, const typename [Field::Element\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, size\_t seuil, size\_t \*x)
- template<class [Field](#) >  
[Field::Element\\_ptr](#) pfgemm\_3D\_rec2 (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t m, const size\_t n, const size\_t k, const typename [Field::Element](#) alpha, const typename [Field::Element\\_ptr](#) A, const size\_t lda, const typename [Field::Element\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, size\_t seuil, size\_t \*x)

- `template<class Field >`  
`Field::Element_ptr fsquare` (const `Field` &F, const `FFLAS_TRANSPOSE` ta, const `size_t` n, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` A, const `size_t` lda, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc)

*fsquare: Squares a matrix.*

### 13.114.1 Macro Definition Documentation

#### 13.114.1.1 \_\_FFLASFFPACK\_fflas\_fflas\_level3\_INL

```
#define __FFLASFFPACK_fflas_fflas_level3_INL
```

#### 13.114.1.2 \_\_FFLAS\_\_TRSM\_READONLY

```
#define __FFLAS__TRSM_READONLY
```

## 13.115 fflas\_pfgemm.inl File Reference

```
#include "fflas-ffpack/paladin/blockcuts.inl"
#include "fflas-ffpack/paladin/parallel.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/paladin/pfgemm_variants.inl"
```

### Namespaces

- namespace `FFLAS`

### Macros

- `#define __FFLASFFPACK_fflas_pfgemm_INL`
- `#define __FFLASFFPACK_SEQPARTHRESHOLD 220`
- `#define __FFLASFFPACK_DIMKPENALTY 1`

### Functions

- `template<class Field , class ModeTrait , class Strat , class Param >`  
`std::enable_if<!std::is_same< ModeTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >::value, typename Field::Element_ptr >::type` `fgemm` (const `Field` &F, const `FFLAS::FFLAS_TRANSPOSE` ta, const `FFLAS::FFLAS_TRANSPOSE` tb, const `size_t` m, const `size_t` n, const `size_t` k, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` A, const `size_t` lda, typename `Field::ConstElement_ptr` B, const `size_t` ldb, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, `MMHelper< Field, MMHelperAlgo::Winograd, ModeTrait, ParSeqHelper::Parallel< Strat, Param > > &H)`

### 13.115.1 Macro Definition Documentation

#### 13.115.1.1 \_\_FFLASFFPACK\_fflas\_pfgemm\_INL

```
#define __FFLASFFPACK_fflas_pfgemm_INL
```

#### 13.115.1.2 \_\_FFLASFFPACK\_SEQPARTHRESHOLD

```
#define __FFLASFFPACK_SEQPARTHRESHOLD 220
```

#### 13.115.1.3 \_\_FFLASFFPACK\_DIMKPENALTY

```
#define __FFLASFFPACK_DIMKPENALTY 1
```

## 13.116 fflas\_pftrsm.inl File Reference

```
#include "fflas-ffpack/paladin/parallel.h"
```

### Namespaces

- namespace [FFLAS](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_pftrsm\\_INL](#)
- #define [PTRSM\\_HYBRID\\_THRESHOLD](#) 256

### Functions

- template<class [Field](#), class Cut, class Param >  
[Field::Element\\_ptr](#) [ftrsm](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const [FFLAS::FFLAS\\_UPLO](#) UpLo, const [FFLAS::FFLAS\\_TRANSPOSE](#) TA, const [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t m, const size\_t n, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) B, const size\_t ldb, [TRSMHelper](#)< [StructureHelper::Iterative](#), [ParSeqHelper::Parallel](#)< Cut, Param > > &H)
- template<class [Field](#), class Cut, class Param >  
[Field::Element\\_ptr](#) [ftrsm](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const [FFLAS::FFLAS\\_UPLO](#) UpLo, const [FFLAS::FFLAS\\_TRANSPOSE](#) TA, const [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t m, const size\_t n, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) B, const size\_t ldb, [TRSMHelper](#)< [StructureHelper::Hybrid](#), [ParSeqHelper::Parallel](#)< Cut, Param > > &H)

### 13.116.1 Macro Definition Documentation

#### 13.116.1.1 [\\_\\_FFLASFFPACK\\_fflas\\_pftrsm\\_INL](#)

```
#define __FFLASFFPACK_fflas_pftrsm_INL
```

#### 13.116.1.2 [PTRSM\\_HYBRID\\_THRESHOLD](#)

```
#define PTRSM_HYBRID_THRESHOLD 256
```

## 13.117 fflas\_simd.h File Reference

```
#include "fflas-ffpack/utils/fflas_intrinsic.h"
#include <iostream>
#include <type_traits>
#include <limits>
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/utils/debug.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include <vector>
#include "givaro/givtypestring.h"
#include <fflas-ffpack/fflas/fflas_simd/simd_modular.inl>
```

### Data Structures

- struct [support\\_simd](#)< T >
- struct [is\\_simd](#)< T >
- struct [NoSimd](#)< T >

- struct [SimdChooser](#)< T, bool, bool >
- struct [SimdChooser](#)< T, false, b >
- struct [SimdChooser](#)< T, true, false >
- struct [SimdChooser](#)< T, true, true >

## Namespaces

- namespace [FFLAS](#)

## Macros

- `#define` [SIMD\\_INT](#) 1
- `#define` [INLINE](#) inline
- `#define` [CONST](#)
- `#define` [PURE](#)
- `#define` [NORML\\_MOD](#)(C, P, NEGP, MIN, MAX, Q, T)
- `#define` [FLOAT\\_MOD](#)(C, P, INVP, Q)

## Typedefs

- `template<class T >`  
using [Simd](#) = typename [SimdChooser](#)<T>::value

## 13.117.1 Macro Definition Documentation

### 13.117.1.1 SIMD\_INT

```
#define SIMD_INT 1
```

### 13.117.1.2 INLINE

```
#define INLINE inline
```

### 13.117.1.3 CONST

```
#define CONST
```

### 13.117.1.4 PURE

```
#define PURE
```

### 13.117.1.5 NORML\_MOD

```
#define NORML_MOD(  
    C,  
    P,  
    NEGP,  
    MIN,  
    MAX,  
    Q,  
    T)
```

#### Value:

```
{  
    \  
    Q = greater(C, MAX);  
    \  
    T = lesser(C, MIN);  
    \  
    Q = vand(Q, NEGP);  
    \  
    T = vand(T, P);  
    \  
}
```

```

    Q = vor(Q, T);
    \
    C = add(C, Q);
    \
}

```

### 13.117.1.6 FLOAT\_MOD

```

#define FLOAT_MOD(
    C,
    P,
    INVP,
    Q)

```

**Value:**

```

{
    \
    Q = mul(C, INVP);
    \
    Q = floor(Q);
    \
    C = fnmadd(C, Q, P);
    \
}

```

## 13.117.2 Typedef Documentation

### 13.117.2.1 Simd

```

template<class T >
using Simd = typename SimdChooser<T>::value

```

## 13.118 simd.doxy File Reference

## 13.119 simd128.inl File Reference

```

#include "simd128_float.inl"
#include "simd128_double.inl"

```

### Data Structures

- struct [Simd128i\\_base](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_ffpack\\_utils\\_simd128\\_INL](#)

### Typedefs

- template<class T >  
using [Simd128](#)

## 13.119.1 Macro Definition Documentation

### 13.119.1.1 \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_INL

```

#define __FFLASFFPACK_fflas_ffpack_utils_simd128_INL

```

## 13.119.2 Typedef Documentation

### 13.119.2.1 Simd128

```

template<class T >
using Simd128

```

**Initial value:**

```
Simd128_impl<std::is_arithmetic<T>::value, std::is_integral<T>::value, std::is_signed<T>::value, sizeof(T)>
```

## 13.120 simd128\_double.inl File Reference

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include <vector>
#include <type_traits>
```

**Data Structures**

- struct [Simd128\\_impl< true, false, true, 8 >](#)

**Macros**

- [#define \\_\\_FFLASFFPACK\\_fflas\\_ffpack\\_utils\\_simd128\\_double\\_INL](#)

### 13.120.1 Macro Definition Documentation

#### 13.120.1.1 \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_double\_INL

```
#define __FFLASFFPACK_fflas_ffpack_utils_simd128_double_INL
```

## 13.121 simd128\_float.inl File Reference

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include <vector>
#include <type_traits>
```

**Data Structures**

- struct [Simd128\\_impl< true, false, true, 4 >](#)

**Macros**

- [#define \\_\\_FFLASFFPACK\\_fflas\\_ffpack\\_utils\\_simd128\\_float\\_INL](#)

### 13.121.1 Macro Definition Documentation

#### 13.121.1.1 \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_float\_INL

```
#define __FFLASFFPACK_fflas_ffpack_utils_simd128_float_INL
```

## 13.122 simd128\_int16.inl File Reference

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include <vector>
#include <type_traits>
```



**Data Structures**

- struct [Simd128\\_impl< true, true, true, 2 >](#)
- union [Simd128\\_impl< true, true, true, 2 >::Converter](#)
- struct [Simd128\\_impl< true, true, false, 2 >](#)
- union [Simd128\\_impl< true, true, false, 2 >::Converter](#)

**Macros**

- `#define __FFLASFFPACK_fflas_ffpack_utils_simd128_int16_INL`

**13.122.1 Macro Definition Documentation****13.122.1.1 \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_int16\_INL**

```
#define __FFLASFFPACK_fflas_ffpack_utils_simd128_int16_INL
```

**13.123 simd128\_int32.inl File Reference**

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include <vector>
#include <type_traits>
```

**Data Structures**

- struct [Simd128\\_impl< true, true, true, 4 >](#)
- union [Simd128\\_impl< true, true, true, 4 >::Converter](#)
- struct [Simd128\\_impl< true, true, false, 4 >](#)
- union [Simd128\\_impl< true, true, false, 4 >::Converter](#)

**Macros**

- `#define __FFLASFFPACK_fflas_ffpack_utils_simd128_int32_INL`

**13.123.1 Macro Definition Documentation****13.123.1.1 \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_int32\_INL**

```
#define __FFLASFFPACK_fflas_ffpack_utils_simd128_int32_INL
```

**13.124 simd128\_int64.inl File Reference**

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include "fflas-ffpack/utils/bit_manipulation.h"
#include <vector>
#include <type_traits>
```

**Data Structures**

- struct [Simd128\\_impl< true, true, true, 8 >](#)
- union [Simd128\\_impl< true, true, true, 8 >::Converter](#)
- struct [Simd128\\_impl< true, true, false, 8 >](#)
- union [Simd128\\_impl< true, true, false, 8 >::Converter](#)

**Macros**

- `#define __FFLASFFPACK_fflas_ffpack_utils_simd128_int64_INL`
- `#define vect_t Simd128_impl<true,true,true,8>::vect_t`

**13.124.1 Macro Definition Documentation****13.124.1.1 \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_int64\_INL**

```
#define __FFLASFFPACK_fflas_ffpack_utils_simd128_int64_INL
```

**13.124.1.2 vect\_t**

```
#define vect_t Simd128_impl<true,true,true,8>::vect_t
```

**13.125 simd256.inl File Reference**

```
#include "simd256_float.inl"
#include "simd256_double.inl"
```

**Data Structures**

- struct [Simd256fp\\_base](#)
- struct [Simd256i\\_base](#)

**Macros**

- `#define __FFLASFFPACK_fflas_ffpack_utils_simd256_INL`

**Typedefs**

- `template<class T >`  
`using Simd256`

**13.125.1 Macro Definition Documentation****13.125.1.1 \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_INL**

```
#define __FFLASFFPACK_fflas_ffpack_utils_simd256_INL
```

**13.125.2 Typedef Documentation****13.125.2.1 Simd256**

```
template<class T >
using Simd256
```

**Initial value:**

```
Simd256\_impl<std::is_arithmetic<T>::value, std::is_integral<T>::value, std::is_signed<T>::value, sizeof(T)>
```

**13.126 simd256\_double.inl File Reference**

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include <vector>
#include <type_traits>
```

## Data Structures

- struct [Simd256\\_impl< true, false, true, 8 >](#)
- union [Simd256\\_impl< true, false, true, 8 >::Converter](#)

## Macros

- `#define __FFLASFFPACK_fflas_ffpack_utils_simd256_double_INL`

### 13.126.1 Macro Definition Documentation

#### 13.126.1.1 \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_double\_INL

```
#define __FFLASFFPACK_fflas_ffpack_utils_simd256_double_INL
```

## 13.127 simd256\_float.inl File Reference

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include <vector>
#include <type_traits>
```

## Data Structures

- struct [Simd256\\_impl< true, false, true, 4 >](#)

## Macros

- `#define __FFLASFFPACK_fflas_ffpack_utils_simd256_float_INL`

### 13.127.1 Macro Definition Documentation

#### 13.127.1.1 \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_float\_INL

```
#define __FFLASFFPACK_fflas_ffpack_utils_simd256_float_INL
```

## 13.128 simd256\_int16.inl File Reference

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include <vector>
#include <type_traits>
```

## Data Structures

- struct [Simd256\\_impl< true, true, true, 2 >](#)
- union [Simd256\\_impl< true, true, true, 2 >::Converter](#)
- struct [Simd256\\_impl< true, true, false, 2 >](#)
- union [Simd256\\_impl< true, true, false, 2 >::Converter](#)

## Macros

- `#define __FFLASFFPACK_fflas_ffpack_utils_simd256_int16_INL`

### 13.128.1 Macro Definition Documentation

#### 13.128.1.1 \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_int16\_INL

```
#define __FFLASFFPACK_fflas_ffpack_utils_simd256_int16_INL
```

### 13.129 simd256\_int32.inl File Reference

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include <vector>
#include <type_traits>
```

#### Data Structures

- struct [Simd256\\_impl< true, true, true, 4 >](#)
- union [Simd256\\_impl< true, true, true, 4 >::Converter](#)
- struct [Simd256\\_impl< true, true, false, 4 >](#)
- union [Simd256\\_impl< true, true, false, 4 >::Converter](#)

#### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_ffpack\\_utils\\_simd256\\_int32\\_INL](#)

### 13.129.1 Macro Definition Documentation

#### 13.129.1.1 \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_int32\_INL

```
#define __FFLASFFPACK_fflas_ffpack_utils_simd256_int32_INL
```

### 13.130 simd256\_int64.inl File Reference

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include "fflas-ffpack/utils/bit_manipulation.h"
#include <vector>
#include <type_traits>
```

#### Data Structures

- struct [Simd256\\_impl< true, true, true, 8 >](#)
- union [Simd256\\_impl< true, true, true, 8 >::Converter](#)
- struct [Simd256\\_impl< true, true, false, 8 >](#)
- union [Simd256\\_impl< true, true, false, 8 >::Converter](#)

#### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_ffpack\\_utils\\_simd256\\_int64\\_INL](#)
- #define [vect\\_t Simd256\\_impl<true, true, true, 8>::vect\\_t](#)

### 13.130.1 Macro Definition Documentation

#### 13.130.1.1 \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_int64\_INL

```
#define __FFLASFFPACK_fflas_ffpack_utils_simd256_int64_INL
```

### 13.130.1.2 vect\_t

```
#define vect_t Simd256_impl<true, true, true, 8>::vect_t
```

## 13.131 simd512.inl File Reference

```
#include "simd512_float.inl"
#include "simd512_double.inl"
#include "simd512_int64.inl"
```

### Data Structures

- struct [Simd512i\\_base](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_simd512\\_INL](#)

### Typedefs

- template<class T >  
using [Simd512](#)

## 13.131.1 Macro Definition Documentation

### 13.131.1.1 \_\_FFLASFFPACK\_simd512\_INL

```
#define __FFLASFFPACK_simd512_INL
```

## 13.131.2 Typedef Documentation

### 13.131.2.1 Simd512

```
template<class T >
using Simd512
```

#### Initial value:

```
Simd512_impl<std::is_arithmetic<T>::value, std::is_integral<T>::value, std::is_signed<T>::value, sizeof(T)>
```

## 13.132 simd512\_double.inl File Reference

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include <vector>
#include <type_traits>
```

### Data Structures

- struct [Simd512\\_impl< true, false, true, 8 >](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_simd512\\_double\\_INL](#)

### 13.132.1 Macro Definition Documentation

#### 13.132.1.1 \_\_FFLASFFPACK\_simd512\_double\_INL

```
#define __FFLASFFPACK_simd512_double_INL
```

### 13.133 simd512\_float.inl File Reference

```
#include "givaro/givtypestring.h"  
#include "fflas-ffpack/utils/align-allocator.h"  
#include <vector>  
#include <type_traits>
```

#### Data Structures

- struct [Simd512\\_impl](#)< true, false, true, 4 >

#### Macros

- #define [\\_\\_FFLASFFPACK\\_simd512\\_float\\_INL](#)

### 13.133.1 Macro Definition Documentation

#### 13.133.1.1 \_\_FFLASFFPACK\_simd512\_float\_INL

```
#define __FFLASFFPACK_simd512_float_INL
```

### 13.134 simd512\_int32.inl File Reference

```
#include "fflas-ffpack/fflas/fflas_simd/simd512_int64.inl"  
#include "givaro/givtypestring.h"  
#include "fflas-ffpack/utils/align-allocator.h"  
#include <vector>  
#include <type_traits>
```

#### Data Structures

- struct [Simd256\\_impl](#)< true, true, true, 4 >
- union [Simd256\\_impl](#)< true, true, true, 4 >::Converter
- struct [Simd256\\_impl](#)< true, true, false, 4 >
- union [Simd256\\_impl](#)< true, true, false, 4 >::Converter

#### Macros

- #define [\\_\\_FFLASFFPACK\\_simd512\\_int32\\_INL](#)

### 13.134.1 Macro Definition Documentation

#### 13.134.1.1 \_\_FFLASFFPACK\_simd512\_int32\_INL

```
#define __FFLASFFPACK_simd512_int32_INL
```

## 13.135 simd512\_int64.inl File Reference

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include "fflas-ffpack/utils/bit_manipulation.h"
#include <vector>
#include <type_traits>
```

### Data Structures

- struct [Simd512\\_impl< true, true, true, 8 >](#)
- union [Simd512\\_impl< true, true, true, 8 >::Converter](#)
- struct [Simd512\\_impl< true, true, false, 8 >](#)
- union [Simd512\\_impl< true, true, false, 8 >::Converter](#)

### Macros

- [#define \\_simd512\\_int64\\_INL](#)
- [#define vect\\_t Simd512\\_impl<true, true, true, 8>::vect\\_t](#)

### 13.135.1 Macro Definition Documentation

#### 13.135.1.1 \_simd512\_int64\_INL

```
#define _simd512_int64_INL
```

#### 13.135.1.2 vect\_t

```
#define vect_t Simd512_impl<true, true, true, 8>::vect_t
```

## 13.136 simd\_modular.inl File Reference

### Data Structures

- class [FieldSimd< \\_Field >](#)

## 13.137 fflas\_sparse.h File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/config.h"
#include "fflas-ffpack/config-blas.h"
#include "fflas-ffpack/paladin/parallel.h"
#include <recint/recint.h>
#include <givaro/udl.h>
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/field/field-traits.h"
#include "fflas-ffpack/fflas/fflas_bounds.inl"
#include "fflas-ffpack/utils/fflas_memory.h"
#include <type_traits>
#include <vector>
#include <iostream>
#include "fflas-ffpack/fflas/fflas_sparse/sparse_matrix_traits.h"
#include "fflas-ffpack/fflas/fflas_sparse/utils.h"
#include "fflas-ffpack/fflas/fflas_sparse/csr.h"
#include "fflas-ffpack/fflas/fflas_sparse/coo.h"
#include "fflas-ffpack/fflas/fflas_sparse/ell.h"
#include "fflas-ffpack/fflas/fflas_sparse/sell.h"
```

```
#include "fflas-ffpack/fflas/fflas_sparse/csr_hyb.h"
#include "fflas-ffpack/fflas/fflas_sparse/ell_simd.h"
#include "fflas-ffpack/fflas/fflas_sparse/hyb_zo.h"
#include "fflas-ffpack/fflas/fflas_sparse.inl"
#include "fflas-ffpack/fflas/fflas_sparse/read_sparse.h"
```

## Data Structures

- struct [HelperFlag](#)
- struct [CsrMat< Field >](#)
- struct [CooMat< Field >](#)
- struct [EllMat< Field >](#)
- struct [SpMat< Field, flag >](#)

## Namespaces

- namespace [MKL\\_CONFIG](#)
- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details](#)

## Macros

- #define [index\\_t](#) uint32\_t
- #define [ROUND\\_DOWN](#)(x, s)
- #define [\\_\\_FFLASFFPACK\\_CACHE\\_LINE\\_SIZE](#) 64
- #define [assume\\_aligned](#)(pout, pin, v)
- #define [DENSE\\_THRESHOLD](#) 0.5

## Enumerations

- enum class [SparseMatrix\\_t](#) {  
[CSR](#) , [CSR\\_ZO](#) , [CSC](#) , [CSC\\_ZO](#) ,  
[COO](#) , [COO\\_ZO](#) , [ELL](#) , [ELL\\_ZO](#) ,  
[SELL](#) , [SELL\\_ZO](#) , [ELL\\_simd](#) , [ELL\\_simd\\_ZO](#) ,  
[CSR\\_HYB](#) , [HYB\\_ZO](#) }

## Functions

- template<class [Field](#) >  
void [init\\_y](#) (const [Field](#) &F, const size\_t m, const typename [Field::Element](#) b, typename [Field::Element\\_ptr](#) y)
- template<class [Field](#) >  
void [init\\_y](#) (const [Field](#) &F, const size\_t m, const size\_t n, const typename [Field::Element](#) b, typename [Field::Element\\_ptr](#) y, const int ldy)
- template<class [Field](#) , class SM , class FC , class MZO >  
std::enable\_if<!(std::is\_same< typenameElementTraits< typenameField::Element >::value, [ElementCategories::MachineFloat](#)  
>::value||std::is\_same< typenameElementTraits< typenameField::Element >::value, [ElementCategories::MachineIntTag](#)  
>::value)>::type [fspmv\\_dispatch](#) (const [Field](#) &F, const SM &A, typename [Field::ConstElement\\_ptr](#) x, type-  
name [Field::Element\\_ptr](#) y, FC fc, MZO mzo)
- template<class [Field](#) , class SM , class FC , class MZO >  
std::enable\_if< std::is\_same< typenameElementTraits< typenameField::Element >::value, [ElementCategories::MachineFloat](#)  
>::value||std::is\_same< typenameElementTraits< typenameField::Element >::value, [ElementCategories::MachineIntTag](#)  
>::value >::type [fspmv\\_dispatch](#) (const [Field](#) &F, const SM &A, typename [Field::ConstElement\\_ptr](#) x, type-  
name [Field::Element\\_ptr](#) y, FC fc, MZO mzo)
- template<class [Field](#) , class SM >  
void [fspmv](#) (const [Field](#) &F, const SM &A, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#)  
y, [FieldCategories::GenericTag](#), [NotZOSparseMatrix](#))



- `template<class Field , class SM >`  
`std::enable_if< !isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< !isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`void fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< !isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`void fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, std::true_type)`
- `template<class Field , class SM , class FCat , class MZO >`  
`std::enable_if< !(std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value) || std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value) >::type fspmm_dispatch (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FCat, MZO)`
- `template<class Field , class SM , class FCat , class MZO >`  
`std::enable_if< std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value) || std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value >::type fspmm_dispatch (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FCat, MZO)`
- `template<class Field , class SM >`  
`void fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< !support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::ModularTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< !support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM`

- &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x, int ldx, typename [Field::Element\\_ptr](#) y, int ldy, [FieldCategories::ModularTag](#), [NotZOSparseMatrix](#))
- `template<class Field , class SM >`  
`void fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::GenericTag, ZOSparseMatrix)`
  - `template<class Field , class SM >`  
`std::enable_if< support\_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::UnparametricTag, ZOSparseMatrix)`
  - `template<class Field , class SM >`  
`std::enable_if<!support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::UnparametricTag, ZOSparseMatrix)`
  - `template<class Field , class SM >`  
`void fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::ModularTag, ZOSparseMatrix)`
  - `template<class Field , class SM , class FCat , class MZO >`  
`std::enable_if<!(std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value)||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value)>::type pfspmm_dispatch (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FCat, MZO)`
  - `template<class Field , class SM , class FCat , class MZO >`  
`std::enable_if< std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value)||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value >::type pfspmm_dispatch (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FCat, MZO)`
  - `template<class Field , class SM >`  
`void pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::GenericTag, NotZOSparseMatrix)`
  - `template<class Field , class SM >`  
`std::enable_if< support\_simd< typenameField::Element >::value >::type pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
  - `template<class Field , class SM >`  
`std::enable_if<!support_simd< typenameField::Element >::value >::type pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
  - `template<class Field , class SM >`  
`std::enable_if< support\_simd< typenameField::Element >::value >::type pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::ModularTag, NotZOSparseMatrix)`
  - `template<class Field , class SM >`  
`std::enable_if<!support_simd< typenameField::Element >::value >::type pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::ModularTag, NotZOSparseMatrix)`
  - `template<class Field , class SM >`  
`void pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::GenericTag, ZOSparseMatrix)`
  - `template<class Field , class SM >`  
`std::enable_if< support\_simd< typenameField::Element >::value >::type pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::UnparametricTag, ZOSparseMatrix)`
  - `template<class Field , class SM >`  
`std::enable_if<!support_simd< typenameField::Element >::value >::type pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::UnparametricTag, ZOSparseMatrix)`

- `template<class Field, class SM >`  
`void pfpmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::ModularTag, ZOSparseMatrix)`
- `template<class Field, class SM >`  
`void pfpmmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag, std::false_type)`
- `template<class Field, class SM >`  
`void pfpmmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, std::false_type)`
- `template<class Field, class SM >`  
`void pfpmmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, std::false_type)`
- `template<class Field, class SM >`  
`void pfpmmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag, std::true_type)`
- `template<class Field, class SM >`  
`void pfpmmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, std::true_type)`
- `template<class Field, class SM >`  
`void pfpmmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, std::true_type)`
- `template<class Field, class SM >`  
`void fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, const typename Field::Element &beta, typename Field::Element_ptr y)`
- `template<class Field, class SM >`  
`void fspm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, const typename Field::Element &beta, typename Field::Element_ptr y, int ldy)`

## 13.137.1 Macro Definition Documentation

### 13.137.1.1 index\_t

```
#define index_t uint32_t
```

### 13.137.1.2 ROUND\_DOWN

```
#define ROUND_DOWN(  
    x,  
    s)
```

#### Value:

```
((x) & ~( (s)-1))
```

### 13.137.1.3 \_\_FFLASFFPACK\_CACHE\_LINE\_SIZE

```
#define __FFLASFFPACK_CACHE_LINE_SIZE 64
```

### 13.137.1.4 assume\_aligned

```
#define assume_aligned(  
    pout,  
    pin,  
    v)
```

#### Value:

```
decltype(pin) pout = pin;
```

### 13.137.1.5 DENSE\_THRESHOLD

```
#define DENSE_THRESHOLD 0.5
```

## 13.138 fflas\_sparse.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details](#)

### Macros

- `#define` [\\_\\_FFLASFFPACK\\_fflas\\_fflas\\_sparse\\_INL](#)

### Functions

- `template<class Field >`  
void [init\\_y](#) (const [Field](#) &F, const size\_t m, const typename [Field::Element](#) b, typename [Field::Element\\_ptr](#) y)
- `template<class Field >`  
void [init\\_y](#) (const [Field](#) &F, const size\_t m, const size\_t n, const typename [Field::Element](#) b, typename [Field::Element\\_ptr](#) y, const int ldy)
- `template<class Field , class SM , class FC , class MZO >`  
std::enable\_if<!std::is\_same< typenameElementTraits< typenameField::Element >::value, [ElementCategories::MachineFloat](#) >::value||std::is\_same< typenameElementTraits< typenameField::Element >::value, [ElementCategories::MachineIntTag](#) >::value>::type [fspmvp\\_dispatch](#) (const [Field](#) &F, const SM &A, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, FC fc, MZO mzo)
- `template<class Field , class SM , class FC , class MZO >`  
std::enable\_if< std::is\_same< typenameElementTraits< typenameField::Element >::value, [ElementCategories::MachineFloat](#) >::value||std::is\_same< typenameElementTraits< typenameField::Element >::value, [ElementCategories::MachineIntTag](#) >::value >::type [fspmvp\\_dispatch](#) (const [Field](#) &F, const SM &A, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, FC fc, MZO mzo)
- `template<class Field , class SM >`  
void [fspmvp](#) (const [Field](#) &F, const SM &A, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, [FieldCategories::GenericTag](#), [NotZOSparseMatrix](#))
- `template<class Field , class SM >`  
std::enable\_if<[isSparseMatrixSimdFormat](#)< [Field](#), SM >::value >::type [fspmvp](#) (const [Field](#) &F, const SM &A, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, [FieldCategories::UnparametricTag](#), [NotZOSparseMatrix](#))
- `template<class Field , class SM >`  
std::enable\_if< [isSparseMatrixSimdFormat](#)< [Field](#), SM >::value &&[support\\_simd](#)< typenameField::Element >::value >::type [fspmvp](#) (const [Field](#) &F, const SM &A, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, [FieldCategories::UnparametricTag](#), [NotZOSparseMatrix](#))
- `template<class Field , class SM >`  
std::enable\_if<[isSparseMatrixSimdFormat](#)< [Field](#), SM >::value >::type [fspmvp](#) (const [Field](#) &F, const SM &A, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, [FieldCategories::ModularTag](#), [NotZOSparseMatrix](#))
- `template<class Field , class SM >`  
std::enable\_if< [isSparseMatrixSimdFormat](#)< [Field](#), SM >::value &&[support\\_simd](#)< typenameField::Element >::value >::type [fspmvp](#) (const [Field](#) &F, const SM &A, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, [FieldCategories::ModularTag](#), [NotZOSparseMatrix](#))
- `template<class Field , class SM >`  
void [fspmvp](#) (const [Field](#) &F, const SM &A, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, [FieldCategories::GenericTag](#), [ZOSparseMatrix](#))
- `template<class Field , class SM >`  
std::enable\_if<[isSparseMatrixSimdFormat](#)< [Field](#), SM >::value >::type [fspmvp](#) (const [Field](#) &F, const SM &A, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, [FieldCategories::UnparametricTag](#), [ZOSparseMatrix](#))
- `template<class Field , class SM >`  
std::enable\_if< [isSparseMatrixSimdFormat](#)< [Field](#), SM >::value &&[support\\_simd](#)< typenameField::Element >::value >::type [fspmvp](#) (const [Field](#) &F, const SM &A, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, [FieldCategories::UnparametricTag](#), [ZOSparseMatrix](#))

- `template<class Field , class SM >`  
`void fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, std::true_type)`
- `template<class Field , class SM , class FCat , class MZO >`  
`std::enable_if<!(std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value)||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value)>::type fspmm_dispatch (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FCat, MZO)`
- `template<class Field , class SM , class FCat , class MZO >`  
`std::enable_if< std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value)||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value >::type fspmm_dispatch (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FCat, MZO)`
- `template<class Field , class SM >`  
`void fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if<!support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::ModularTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if<!support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::ModularTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`void fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if<!support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`void fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::ModularTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`void fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, const typename Field::Element &beta, typename Field::Element_ptr y)`
- `template<class Field , class SM >`  
`void fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, const typename Field::Element &beta, typename Field::Element_ptr y, int ldy)`

## 13.138.1 Macro Definition Documentation

### 13.138.1.1 \_\_FFLASFFPACK\_fflas\_fflas\_sparse\_INL

```
#define __FFLASFFPACK_fflas_fflas_sparse_INL
```

## 13.139 coo.h File Reference

```
#include "fflas-ffpack/fflas/fflas_sparse/coo/coo_utils.inl"
#include "fflas-ffpack/fflas/fflas_sparse/coo/coo_spmv.inl"
#include "fflas-ffpack/fflas/fflas_sparse/coo/coo_spmmm.inl"
```

### Data Structures

- struct [Sparse< \\_Field, SparseMatrix\\_t::COO >](#)
- struct [Sparse< \\_Field, SparseMatrix\\_t::COO\\_ZO >](#)

### Namespaces

- namespace [FFLAS](#)

### Functions

- template<class [Field](#) , class [IndexT](#) >  
void [sparse\\_init](#) (const [Field](#) &F, [Sparse< Field, SparseMatrix\\_t::COO >](#) &A, const [IndexT](#) \*row, const [IndexT](#) \*col, typename [Field::ConstElement\\_ptr](#) dat, [uint64\\_t](#) rowdim, [uint64\\_t](#) coldim, [uint64\\_t](#) nnz)
- template<class [Field](#) , class [IndexT](#) >  
void [sparse\\_init](#) (const [Field](#) &F, [Sparse< Field, SparseMatrix\\_t::COO\\_ZO >](#) &A, const [IndexT](#) \*row, const [IndexT](#) \*col, typename [Field::ConstElement\\_ptr](#) dat, [uint64\\_t](#) rowdim, [uint64\\_t](#) coldim, [uint64\\_t](#) nnz)
- template<class [Field](#) >  
void [sparse\\_delete](#) (const [Sparse< Field, SparseMatrix\\_t::COO >](#) &A)
- template<class [Field](#) >  
void [sparse\\_delete](#) (const [Sparse< Field, SparseMatrix\\_t::COO\\_ZO >](#) &A)

## 13.140 coo\_spmmm.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_coo\\_spmmm\\_INL](#)

### Functions

- template<class [Field](#) >  
void [fspmm](#) (const [Field](#) &F, const [Sparse< Field, SparseMatrix\\_t::COO >](#) &A, [size\\_t](#) blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [fspmm](#) (const [Field](#) &F, const [Sparse< Field, SparseMatrix\\_t::COO >](#) &A, [size\\_t](#) blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [fspmm](#) (const [Field](#) &F, const [Sparse< Field, SparseMatrix\\_t::COO >](#) &A, [size\\_t](#) blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, const [int64\\_t](#) kmax)
- template<class [Field](#) >  
void [fspmm\\_simd\\_aligned](#) (const [Field](#) &F, const [Sparse< Field, SparseMatrix\\_t::COO >](#) &A, [size\\_t](#) blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, const [int64\\_t](#) kmax)
- template<class [Field](#) >  
void [fspmm\\_simd\\_unaligned](#) (const [Field](#) &F, const [Sparse< Field, SparseMatrix\\_t::COO >](#) &A, [size\\_t](#) blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, const [int64\\_t](#) kmax)



- `template<class Field >`  
`void fspmm_one (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm_mone (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm_one_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_one_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_mone_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_mone_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`

### 13.140.1 Macro Definition Documentation

#### 13.140.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_coo\_spmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_coo_spmv_INL
```

## 13.141 coo\_spmv.inl File Reference

### Namespaces

- namespace `FFLAS`
- namespace `FFLAS::sparse_details_impl`

### Macros

- `#define __FFLASFFPACK_fflas_sparse_coo_spmv_INL`

### Functions

- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::COO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::COO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::COO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const uint64_t kmax)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`

- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`

### 13.141.1 Macro Definition Documentation

#### 13.141.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_coo\_spmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_coo_spmv_INL
```

## 13.142 coo\_utils.inl File Reference

### Namespaces

- namespace [FFLAS](#)

### Macros

- `#define __FFLASFFPACK_fflas_sparse_coo_utils_INL`

### Functions

- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::COO > &A)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::COO_ZO > &A)`
- `template<class Field, class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::COO > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field, class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::COO_ZO > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`

### 13.142.1 Macro Definition Documentation

#### 13.142.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_coo\_utils\_INL

```
#define __FFLASFFPACK_fflas_sparse_coo_utils_INL
```

## 13.143 csr.h File Reference

```
#include "fflas-ffpack/fflas/fflas_sparse/csr/csr_utils.inl"
#include "fflas-ffpack/fflas/fflas_sparse/csr/csr_spmv.inl"
#include "fflas-ffpack/fflas/fflas_sparse/csr/csr_spmv.inl"
```

### Data Structures

- struct [Sparse< \\_Field, SparseMatrix\\_t::CSR >](#)
- struct [Sparse< \\_Field, SparseMatrix\\_t::CSR\\_ZO >](#)



## Namespaces

- namespace [FFLAS](#)

## Functions

- template<class [Field](#) , class IndexT >  
void [sparse\\_init](#) (const [Field](#) &F, [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A, const IndexT \*row, const IndexT \*col, typename [Field::ConstElement\\_ptr](#) dat, uint64\_t rowdim, uint64\_t coldim, uint64\_t nnz)
- template<class [Field](#) , class IndexT >  
void [sparse\\_init](#) (const [Field](#) &F, [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A, const IndexT \*row, const IndexT \*col, typename [Field::ConstElement\\_ptr](#) dat, uint64\_t rowdim, uint64\_t coldim, uint64\_t nnz)
- template<class [Field](#) >  
void [sparse\\_delete](#) (const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A)
- template<class [Field](#) >  
void [sparse\\_delete](#) (const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A)

## 13.144 csr\_pspmm.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_CSR\\_pspmm\\_INL](#)

### Functions

- template<class [Field](#) >  
void [pfspmm](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [pfspmm](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [pfspmm](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, const int64\_t kmax)
- template<class [Field](#) >  
void [pfspmm\\_one](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [pfspmm\\_mone](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [pfspmm\\_one](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [pfspmm\\_mone](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, [FieldCategories::UnparametricTag](#))

### 13.144.1 Macro Definition Documentation

#### 13.144.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_pspmm\_INL

```
#define __FFLASFFPACK_fflas_sparse_CSR_pspmm_INL
```

## 13.145 csr\_pspmv.inl File Reference

```
#include <thread>
```

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_CSR\\_pspmv\\_INL](#)

### Functions

- template<class [Field](#) >  
void [pfspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [pfspmv\\_task](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, const [index\\_t](#) iStart, const [index\\_t](#) iStop, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [pfspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [pfspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, const [int64\\_t](#) kmax)
- template<class [Field](#) >  
void [pfspmv\\_one](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [pfspmv\\_mone](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [pfspmv\\_one](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [pfspmv\\_mone](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))

### 13.145.1 Macro Definition Documentation

#### 13.145.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_pspmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_CSR_pspmv_INL
```

## 13.146 csr\_spmv.inl File Reference

### Namespaces

- namespace [FFLAS](#)

- namespace [FFLAS::sparse\\_details\\_impl](#)

## Macros

- `#define __FFLASFFPACK_fflas_sparse_CSR_spmv_INL`

## Functions

- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, index_t blockSize, typename Field::ConstElement_ptr x_, index_t ldx, typename Field::Element_ptr y_, index_t ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, const int64_t kmax)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_one_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_one_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_mone_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_mone_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`

## 13.146.1 Macro Definition Documentation

### 13.146.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_spmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_CSR_spmv_INL
```

## 13.147 csr\_spmv.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- `#define` [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_CSR\\_spmv\\_INL](#)

### Functions

- `template<class Field >`  
void [fspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- `template<class Field >`  
void [fspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- `template<class Field >`  
void [fspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, const int64\_t kmax)
- `template<class Field >`  
void [fspmv\\_one](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- `template<class Field >`  
void [fspmv\\_mone](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- `template<class Field >`  
void [fspmv\\_one](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- `template<class Field >`  
void [fspmv\\_mone](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))

### 13.147.1 Macro Definition Documentation

#### 13.147.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_spmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_CSR_spmv_INL
```

## 13.148 csr\_utils.inl File Reference

### Namespaces

- namespace [FFLAS](#)

### Functions

- `template<class Field >`  
void [sparse\\_delete](#) (const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A)
- `template<class Field >`  
void [sparse\\_delete](#) (const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A)
- `template<class Field >`  
std::ostream & [sparse\\_print](#) (std::ostream &os, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A)
- `template<class IndexT >`  
void [sparse\\_init](#) (const Givaro::Modular< Givaro::Integer > &F, [Sparse](#)< Givaro::Modular< Givaro::Integer >, [SparseMatrix\\_t::CSR](#) > &A, const IndexT \*row, const IndexT \*col, Givaro::Integer \*dat, uint64\_t rowdim, uint64\_t coldim, uint64\_t nnz)

- `template<class IndexT >`  
`void sparse_init (const Givaro::ZRing< Givaro::Integer > &F, Sparse< Givaro::ZRing< Givaro::Integer >, SparseMatrix_t::CSR_ZO > &A, const IndexT *row, const IndexT *col, Givaro::Integer *dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class IndexT, size_t RECINT_SIZE>`  
`void sparse_init (const Givaro::ZRing< RecInt::rmint< RECINT_SIZE > > &F, Sparse< Givaro::ZRing< RecInt::rmint< RECINT_SIZE > >, SparseMatrix_t::CSR_ZO > &A, const IndexT *row, const IndexT *col, typename Givaro::ZRing< RecInt::rmint< RECINT_SIZE > >::Element_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class IndexT, size_t RECINT_SIZE>`  
`void sparse_init (const Givaro::ZRing< RecInt::rmint< RECINT_SIZE > > &F, Sparse< Givaro::ZRing< RecInt::rmint< RECINT_SIZE > >, SparseMatrix_t::CSR > &A, const IndexT *row, const IndexT *col, typename Givaro::ZRing< RecInt::rmint< RECINT_SIZE > >::Element_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field, class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::CSR > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field, class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::CSR_ZO > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`

## 13.149 csr\_hyb.h File Reference

```
#include "fflas-ffpack/fflas/fflas_sparse/csr_hyb/csr_hyb_utils.inl"
#include "fflas-ffpack/fflas/fflas_sparse/csr_hyb/csr_hyb_spmv.inl"
#include "fflas-ffpack/fflas/fflas_sparse/csr_hyb/csr_hyb_spmmm.inl"
```

### Data Structures

- struct `Sparse< _Field, SparseMatrix_t::CSR_HYB >`

### Namespaces

- namespace `FFLAS`

### Functions

- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::CSR_HYB > &A)`
- `template<class Field, class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::CSR_HYB > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`

## 13.150 csr\_hyb\_pspmm.inl File Reference

### Namespaces

- namespace `FFLAS`
- namespace `FFLAS::sparse_details_impl`

### Macros

- `#define __FFLASFFPACK_fflas_sparse_CSR_HYB_pspmm_INL`

## Functions

- `template<class Field >`  
`void pfsppmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, type-`  
`name Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfsppmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, type-`  
`name Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfsppmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, type-`  
`name Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfsppmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, type-`  
`t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy,`  
`FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfsppmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, type-`  
`name Field::ConstElement_ptr x, typename Field::Element_ptr y, const int64_t kmax)`
- `template<class Field >`  
`void pfsppmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, type-`  
`name Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, const int64_t kmax)`

### 13.150.1 Macro Definition Documentation

#### 13.150.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_pspmm\_INL

```
#define __FFLASFFPACK_fflas_sparse_CSR_HYB_pspmm_INL
```

## 13.151 csr\_hyb\_pspmv.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- `#define` [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_CSR\\_HYB\\_pspmv\\_INL](#)

## Functions

- `template<class Field >`  
`void pfsppmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfsppmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfsppmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const int64_t kmax)`

### 13.151.1 Macro Definition Documentation

#### 13.151.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_pspmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_CSR_HYB_pspmv_INL
```

## 13.152 csr\_hyb\_spmv.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- `#define __FFLASFFPACK_fflas_sparse_CSR_HYB_spmv_INL`

### Functions

- `template<class Field >`  
void [fspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_HYB](#) > &A, [size\\_t](#) blockSize, [typename](#) [Field::ConstElement\\_ptr](#) x\_, [int](#) ldx, [typename](#) [Field::Element\\_ptr](#) y\_, [int](#) ldy, [FieldCategories::GenericTag](#))
- `template<class Field >`  
void [fspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_HYB](#) > &A, [size\\_t](#) blockSize, [typename](#) [Field::ConstElement\\_ptr](#) x\_, [int](#) ldx, [typename](#) [Field::Element\\_ptr](#) y\_, [int](#) ldy, [FieldCategories::UnparametricTag](#))
- `template<class Field >`  
void [fspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_HYB](#) > &A, [size\\_t](#) blockSize, [typename](#) [Field::ConstElement\\_ptr](#) x\_, [int](#) ldx, [typename](#) [Field::Element\\_ptr](#) y\_, [int](#) ldy, [const int64\\_t](#) kmax)

### 13.152.1 Macro Definition Documentation

#### 13.152.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_spmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_CSR_HYB_spmv_INL
```

## 13.153 csr\_hyb\_spmv.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- `#define __FFLASFFPACK_fflas_sparse_CSR_HYB_spmv_INL`

### Functions

- `template<class Field >`  
void [fspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_HYB](#) > &A, [typename](#) [Field::ConstElement\\_ptr](#) x\_, [typename](#) [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- `template<class Field >`  
void [fspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_HYB](#) > &A, [typename](#) [Field::ConstElement\\_ptr](#) x\_, [typename](#) [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- `template<class Field >`  
void [fspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_HYB](#) > &A, [typename](#) [Field::ConstElement\\_ptr](#) x\_, [typename](#) [Field::Element\\_ptr](#) y\_, [const uint64\\_t](#) kmax)

### 13.153.1 Macro Definition Documentation

#### 13.153.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_spmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_CSR_HYB_spmv_INL
```

## 13.154 csr\_hyb\_utils.inl File Reference

### Data Structures

- struct [Info](#)
- struct [Coo< ValT, IdxT >](#)

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::csr\\_hyb\\_details](#)

### Macros

- `#define` [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_CSR\\_HYB\\_utils\\_INL](#)

### Functions

- `template<class Field >`  
void [sparse\\_delete](#) (const [Sparse< Field, SparseMatrix\\_t::CSR\\_HYB >](#) &A)
- `template<class Field , class IndexT >`  
void [sparse\\_init](#) (const [Field](#) &F, [Sparse< Field, SparseMatrix\\_t::CSR\\_HYB >](#) &A, const IndexT \*row, const IndexT \*col, typename [Field::ConstElement\\_ptr](#) dat, uint64\_t rowdim, uint64\_t coldim, uint64\_t nnz)

### 13.154.1 Macro Definition Documentation

#### 13.154.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_utils\_INL

```
#define __FFLASFFPACK_fflas_sparse_CSR_HYB_utils_INL
```

## 13.155 ell.h File Reference

```
#include "fflas-ffpack/fflas/fflas_sparse/ell/ell_utils.inl"
#include "fflas-ffpack/fflas/fflas_sparse/ell/ell_spmv.inl"
#include "fflas-ffpack/fflas/fflas_sparse/ell/ell_spmv.inl"
```

### Data Structures

- struct [Sparse< \\_Field, SparseMatrix\\_t::ELL >](#)
- struct [Sparse< \\_Field, SparseMatrix\\_t::ELL\\_ZO >](#)

### Namespaces

- namespace [FFLAS](#)

### Functions

- `template<class Field , class IndexT >`  
void [sparse\\_init](#) (const [Field](#) &F, [Sparse< Field, SparseMatrix\\_t::ELL >](#) &A, const IndexT \*row, const IndexT \*col, typename [Field::ConstElement\\_ptr](#) dat, uint64\_t rowdim, uint64\_t coldim, uint64\_t nnz)
- `template<class Field , class IndexT >`  
void [sparse\\_init](#) (const [Field](#) &F, [Sparse< Field, SparseMatrix\\_t::ELL\\_ZO >](#) &A, const IndexT \*row, const IndexT \*col, typename [Field::ConstElement\\_ptr](#) dat, uint64\_t rowdim, uint64\_t coldim, uint64\_t nnz)
- `template<class Field >`  
void [sparse\\_delete](#) (const [Sparse< Field, SparseMatrix\\_t::ELL >](#) &A)
- `template<class Field >`  
void [sparse\\_delete](#) (const [Sparse< Field, SparseMatrix\\_t::ELL\\_ZO >](#) &A)



## 13.156 ell\_pspmm.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- `#define` [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_ELL\\_pspmm\\_INL](#)

### Functions

- `template<class Field >`  
void [pfspmm](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, [FieldCategories::GenericTag](#))
- `template<class Field >`  
void [pfspmm](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x, int ldx, typename [Field::Element\\_ptr](#) y, int ldy, [FieldCategories::GenericTag](#))
- `template<class Field >`  
void [pfspmm](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, [FieldCategories::UnparametricTag](#))
- `template<class Field >`  
void [pfspmm](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x, int ldx, typename [Field::Element\\_ptr](#) y, int ldy, [FieldCategories::UnparametricTag](#))
- `template<class Field >`  
void [pfspmm](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, const int64\_t kmax)
- `template<class Field >`  
void [pfspmm](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x, int ldx, typename [Field::Element\\_ptr](#) y, int ldy, const int64\_t kmax)
- `template<class Field , class Func >`  
void [pfspmm\\_zo](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_ZO](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, Func &&func)
- `template<class Field , class Func >`  
void [pfspmm\\_zo](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_ZO](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x, int ldx, typename [Field::Element\\_ptr](#) y, int ldy, Func &&func)

### 13.156.1 Macro Definition Documentation

#### 13.156.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_pspmm\_INL

```
#define __FFLASFFPACK_fflas_sparse_ELL_pspmm_INL
```

## 13.157 ell\_pspmv.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- `#define` [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_ELL\\_pspmv\\_INL](#)

## Functions

- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const int64_t kmax)`
- `template<class Field >`  
`void pfspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`

## 13.157.1 Macro Definition Documentation

### 13.157.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_pspmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_ELL_pspmv_INL
```

## 13.158 ell\_spm.inl File Reference

### Namespaces

- namespace `FFLAS`
- namespace `FFLAS::sparse_details_impl`

### Macros

- `#define __FFLASFFPACK_fflas_sparse_ELL_spm_INL`

### Functions

- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, const int64_t kmax)`
- `template<class Field >`  
`void fspmm_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`

- `template<class Field >`  
`void fspmm_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_one_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_one_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_mone_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_mone_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`

## 13.158.1 Macro Definition Documentation

### 13.158.1.1 \_\_FflasFfpack\_fflas\_sparse\_ELL\_spmv\_INL

```
#define __FflasFfpack_fflas_sparse_ELL_spmv_INL
```

## 13.159 ell\_spmv.inl File Reference

### Namespaces

- namespace `FFLAS`
- namespace `FFLAS::sparse_details_impl`

### Macros

- `#define __FflasFfpack_fflas_sparse_ELL_spmv_INL`

### Functions

- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const uint64_t kmax)`

- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`

### 13.159.1 Macro Definition Documentation

#### 13.159.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_spmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_ELL_spmv_INL
```

## 13.160 ell\_utils.inl File Reference

```
#include <vector>
```

### Namespaces

- namespace [FFLAS](#)

### Macros

- `#define` [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_ELL\\_utils\\_INL](#)

### Functions

- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::ELL > &A)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::ELL_ZO > &A)`
- `template<class Field, class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::ELL > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field, class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::ELL_ZO > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`

### 13.160.1 Macro Definition Documentation

#### 13.160.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_utils\_INL

```
#define __FFLASFFPACK_fflas_sparse_ELL_utils_INL
```

## 13.161 ell\_simd.h File Reference

```
#include "fflas-ffpack/fflas/fflas_sparse/ell_simd/ell_simd_utils.inl"
#include "fflas-ffpack/fflas/fflas_sparse/ell_simd/ell_simd_spmv.inl"
```

## Data Structures

- struct [Sparse](#)< [\\_Field](#), [SparseMatrix\\_t::ELL\\_simd](#) >
- struct [Sparse](#)< [\\_Field](#), [SparseMatrix\\_t::ELL\\_simd\\_ZO](#) >

## Namespaces

- namespace [FFLAS](#)

## Functions

- template<class [Field](#) , class [IndexT](#) >  
void [sparse\\_init](#) (const [Field](#) &F, [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd](#) > &A, const [IndexT](#) \*row, const [IndexT](#) \*col, typename [Field::ConstElement\\_ptr](#) dat, [uint64\\_t](#) rowdim, [uint64\\_t](#) coldim, [uint64\\_t](#) nnz)
- template<class [Field](#) , class [IndexT](#) >  
void [sparse\\_init](#) (const [Field](#) &F, [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd\\_ZO](#) > &A, const [IndexT](#) \*row, const [IndexT](#) \*col, typename [Field::ConstElement\\_ptr](#) dat, [uint64\\_t](#) rowdim, [uint64\\_t](#) coldim, [uint64\\_t](#) nnz)
- template<class [Field](#) >  
void [sparse\\_delete](#) (const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd](#) > &A)
- template<class [Field](#) >  
void [sparse\\_delete](#) (const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd\\_ZO](#) > &A)

# 13.162 ell\_simd\_pspmv.inl File Reference

## Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

## Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_ELL\\_simd\\_pspmv\\_INL](#)

## Functions

- template<class [Field](#) >  
void [pfspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [pfspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [pfspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, const [uint64\\_t](#) kmax)
- template<class [Field](#) >  
void [pfspmv\\_one](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [pfspmv\\_mone](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [pfspmv\\_one](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [pfspmv\\_mone](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))

## 13.162.1 Macro Definition Documentation

### 13.162.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_simd\_pspmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_ELL_simd_pspmv_INL
```

## 13.163 ell\_simd\_spmv.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_ELL\\_simd\\_spmv\\_INL](#)

### Functions

- [template<class Field >](#)  
void [fspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- [template<class Field >](#)  
void [fspmv\\_simd](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- [template<class Field >](#)  
void [fspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- [template<class Field >](#)  
void [fspmv\\_simd](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, const uint64\_t kmax)
- [template<class Field >](#)  
void [fspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, const uint64\_t kmax)
- [template<class Field >](#)  
void [fspmv\\_one](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- [template<class Field >](#)  
void [fspmv\\_mone](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- [template<class Field >](#)  
void [fspmv\\_one](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- [template<class Field >](#)  
void [fspmv\\_mone](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- [template<class Field >](#)  
void [fspmv\\_one\\_simd](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- [template<class Field >](#)  
void [fspmv\\_mone\\_simd](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))

## 13.163.1 Macro Definition Documentation

### 13.163.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_simd\_spmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_ELL_simd_spmv_INL
```

## 13.164 ell\_simd\_utils.inl File Reference

### Namespaces

- namespace [FFLAS](#)

### Macros

- `#define __FFLASFFPACK_fflas_sparse_ELL_simd_utils_INL`

### Functions

- `template<class Field >`  
void `sparse_delete` (const `Sparse< Field, SparseMatrix_t::ELL_simd >` &A)
- `template<class Field >`  
void `sparse_delete` (const `Sparse< Field, SparseMatrix_t::ELL_simd_ZO >` &A)
- `template<class Field >`  
void `sparse_print` (const `Sparse< Field, SparseMatrix_t::ELL_simd >` &A)
- `template<class Field, class IndexT >`  
void `sparse_init` (const `Field` &F, `Sparse< Field, SparseMatrix_t::ELL_simd >` &A, const `IndexT *row`, const `IndexT *col`, `typename Field::ConstElement_ptr` dat, `uint64_t rowdim`, `uint64_t coldim`, `uint64_t nnz`)
- `template<class Field, class IndexT >`  
void `sparse_init` (const `Field` &F, `Sparse< Field, SparseMatrix_t::ELL_simd_ZO >` &A, const `IndexT *row`, const `IndexT *col`, `typename Field::ConstElement_ptr` dat, `uint64_t rowdim`, `uint64_t coldim`, `uint64_t nnz`)

### 13.164.1 Macro Definition Documentation

#### 13.164.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_simd\_utils\_INL

```
#define __FFLASFFPACK_fflas_sparse_ELL_simd_utils_INL
```

## 13.165 hyb\_zo.h File Reference

```
#include "fflas-ffpack/fflas/fflas_sparse/hyb_zo/hyb_zo_utils.inl"
#include "fflas-ffpack/fflas/fflas_sparse/hyb_zo/hyb_zo_spmv.inl"
#include "fflas-ffpack/fflas/fflas_sparse/hyb_zo/hyb_zo_spmmm.inl"
```

### Data Structures

- struct `Sparse< _Field, SparseMatrix_t::HYB_ZO >`

### Namespaces

- namespace [FFLAS](#)

## 13.166 hyb\_zo\_pspmm.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- `#define __FFLASFFPACK_fflas_sparse_HYB_ZO_pspmm_INL`

## Functions

- `template<class Field >`  
`void pfsppmm (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, size_t blockSize, type-`  
`name Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfsppmm (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, size_t blockSize, type-`  
`name Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfsppmm (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, size_t blockSize, type-`  
`name Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, uint64_t kmax)`

### 13.166.1 Macro Definition Documentation

#### 13.166.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_pspmm\_INL

```
#define __FFLASFFPACK_fflas_sparse_HYB_ZO_pspmm_INL
```

## 13.167 hyb\_zo\_pspmv.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- `#define __FFLASFFPACK_fflas_sparse_HYB_ZO_pspmv_INL`

## Functions

- `template<class Field >`  
`void pfsppmv (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, typename`  
`Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfsppmv (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, typename`  
`Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfsppmv (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, typename`  
`Field::ConstElement_ptr x, typename Field::Element_ptr y, uint64_t kmax)`

### 13.167.1 Macro Definition Documentation

#### 13.167.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_pspmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_HYB_ZO_pspmv_INL
```

## 13.168 hyb\_zo\_spmv.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- `#define __FFLASFFPACK_fflas_sparse_HYB_ZO_spmv_INL`



## Functions

- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, size_t blockSize, type-`  
`name Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, size_t blockSize, type-`  
`name Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, size_t blockSize, type-`  
`name Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, uint64_t kmax)`

## 13.168.1 Macro Definition Documentation

### 13.168.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_spmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_HYB_ZO_spmv_INL
```

## 13.169 hyb\_zo\_spmv.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- `#define __FFLASFFPACK_fflas_sparse_HYB_ZO_spmv_INL`

## Functions

- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, typename Field::ConstElement_ptr`  
`x, typename Field::Element_ptr y, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, typename Field::ConstElement_ptr`  
`x, typename Field::Element_ptr y, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, typename Field::ConstElement_ptr`  
`x, typename Field::Element_ptr y, uint64_t kmax)`

## 13.169.1 Macro Definition Documentation

### 13.169.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_spmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_HYB_ZO_spmv_INL
```

## 13.170 hyb\_zo\_utils.inl File Reference

### Namespaces

- namespace [FFLAS](#)

### Macros

- `#define __FFLASFFPACK_fflas_sparse_HYB_ZO_utils_INL`

## Functions

- template<class [Field](#) >  
void [sparse\\_delete](#) (const [Sparse](#)< [Field](#), [SparseMatrix\\_t::HYB\\_ZO](#) > &A)
- template<class [Field](#) , class [IndexT](#) >  
void [sparse\\_init](#) (const [Field](#) &F, [Sparse](#)< [Field](#), [SparseMatrix\\_t::HYB\\_ZO](#) > &A, const [IndexT](#) \*row, const [IndexT](#) \*col, typename [Field::ConstElement\\_ptr](#) dat, [uint64\\_t](#) rowdim, [uint64\\_t](#) coldim, [uint64\\_t](#) nnz)
- template<typename [\\_Field](#) >  
std::ostream & [operator<<](#) (std::ostream &os, const [Sparse](#)< [\\_Field](#), [SparseMatrix\\_t::HYB\\_ZO](#) > &A)

## 13.170.1 Macro Definition Documentation

### 13.170.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_utils\_INL

```
#define __FFLASFFPACK_fflas_sparse_HYB_ZO_utils_INL
```

## 13.171 read\_sparse.h File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <fstream>
#include <string>
#include <cstdlib>
#include <iterator>
```

## Data Structures

- struct [Coo](#)< [Field](#) >
- struct [readMyMachineType](#)< [Field](#), [T](#) >
- struct [readMyMachineType](#)< [Field](#), [mpz\\_t](#) >

## Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::details\\_spmv](#)

## Macros

- #define [DNS\\_BIN\\_VER](#) 0
- #define [mask\\_t](#) [uint64\\_t](#)

## Functions

- template<class [Field](#) , bool sorted = true, bool read\_integer = false>  
void [readSmsFormat](#) (const std::string &path, const [Field](#) &f, [index\\_t](#) \*&row, [index\\_t](#) \*&col, typename [Field::Element\\_ptr](#) &val, [index\\_t](#) &rowdim, [index\\_t](#) &coldim, [uint64\\_t](#) &nnz)
- template<class [Field](#) >  
void [readSprFormat](#) (const std::string &path, const [Field](#) &f, [index\\_t](#) \*&row, [index\\_t](#) \*&col, typename [Field::Element\\_ptr](#) &val, [index\\_t](#) &rowdim, [index\\_t](#) &coldim, [uint64\\_t](#) &nnz)
- template<class [T](#) >  
std::enable\_if< std::is\_integral< [T](#) >::value, int > [getDataType](#) ()
- template<class [T](#) >  
std::enable\_if< std::is\_floating\_point< [T](#) >::value, int > [getDataType](#) ()
- template<class [T](#) >  
std::enable\_if< std::is\_same< [T](#), [mpz\\_t](#) >::value, int > [getDataType](#) ()
- template<class [T](#) >  
int [getDataType](#) ()

- template<class [Field](#) >  
void [readMachineType](#) (const [Field](#) &F, typename [Field::Element](#) &modulo, typename [Field::Element\\_ptr](#) val, std::ifstream &file, const uint64\_t dims, const [mask\\_t](#) data\_type, const [mask\\_t](#) field\_desc)
- template<class [Field](#) >  
void [readDnsFormat](#) (const std::string &path, const [Field](#) &F, [index\\_t](#) &rowdim, [index\\_t](#) &colldim, typename [Field::Element\\_ptr](#) &val)
- template<class [Field](#) >  
void [writeDnsFormat](#) (const std::string &path, const [Field](#) &F, const [index\\_t](#) &rowdim, const [index\\_t](#) &colldim, typename [Field::Element\\_ptr](#) A, [index\\_t](#) ldA)

### 13.171.1 Macro Definition Documentation

#### 13.171.1.1 DNS\_BIN\_VER

```
#define DNS_BIN_VER 0
```

#### 13.171.1.2 mask\_t

```
#define mask_t uint64_t
```

## 13.172 sell.h File Reference

```
#include "fflas-ffpack/fflas/fflas_sparse/sell/sell_utils.inl"
#include "fflas-ffpack/fflas/fflas_sparse/sell/sell_spmv.inl"
```

### Data Structures

- struct [Sparse](#)< [\\_Field](#), [SparseMatrix\\_t::SELL](#) >
- struct [Sparse](#)< [\\_Field](#), [SparseMatrix\\_t::SELL\\_ZO](#) >

### Namespaces

- namespace [FFLAS](#)

## 13.173 sell\_pspmv.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_sell\\_pspmv\\_INL](#)

### Functions

- template<class [Field](#) >  
void [pfspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::SELL](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [pfspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::SELL](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [pfspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::SELL](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, const int64\_t kmax)

- `template<class Field >`  
`void pfspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`

### 13.173.1 Macro Definition Documentation

#### 13.173.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_sell\_pspmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_sell_pspmv_INL
```

## 13.174 sell\_spmv.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- `#define __FFLASFFPACK_fflas_sparse_sell_spmv_INL`

### Functions

- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::SELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_simd (const Field &F, const Sparse< Field, SparseMatrix_t::SELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::SELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_simd (const Field &F, const Sparse< Field, SparseMatrix_t::SELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const uint64_t kmax)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::SELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const uint64_t kmax)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_one_simd (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`

- `template<class Field >`  
`void fspmv_mone_simd (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`

### 13.174.1 Macro Definition Documentation

#### 13.174.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_sell\_spmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_sell_spmv_INL
```

## 13.175 sell\_utils.inl File Reference

### Data Structures

- struct [Info](#)
- struct [Coo< ValT, IdxT >](#)

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sell\\_details](#)

### Macros

- `#define __FFLASFFPACK_fflas_sparse_sell_utils_INL`

### Functions

- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::SELL > &A)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::SELL_ZO > &A)`
- `template<class Field >`  
`void sparse_print (const Sparse< Field, SparseMatrix_t::SELL > &A)`
- `template<class Field, class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::SELL > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz, uint64_t sigma=0)`
- `template<class Field, class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::SELL_ZO > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`

### 13.175.1 Macro Definition Documentation

#### 13.175.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_sell\_utils\_INL

```
#define __FFLASFFPACK_fflas_sparse_sell_utils_INL
```

## 13.176 `sparse_matrix_traits.h` File Reference

```
#include <type_traits>
```

### Data Structures

- struct [isSparseMatrix](#)< Field, M >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::CSR > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::CSR\_ZO > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::COO > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::COO\_ZO > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::ELL > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::ELL\_ZO > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::SELL > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::SELL\_ZO > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::ELL\_simd > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::ELL\_simd\_ZO > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::CSR\_HYB > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::HYB\_ZO > >
- struct [isZOSparseMatrix](#)< F, M >
- struct [isZOSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::CSR\_ZO > >
- struct [isZOSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::COO\_ZO > >
- struct [isZOSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::ELL\_ZO > >
- struct [isZOSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::SELL\_ZO > >
- struct [isZOSparseMatrix](#)< Field, Sparse< Field, SparseMatrix\_t::ELL\_simd\_ZO > >
- struct [isSparseMatrixSimdFormat](#)< F, M >
- struct [isSparseMatrixMKLFormat](#)< F, M >
- struct [tfn\\_plus](#)
- struct [tfn\\_mul](#)
- struct [tfn\\_mul\\_eq](#)
- struct [tfn\\_minus](#)
- struct [tfn\\_plus\\_eq](#)
- struct [tfn\\_minus\\_eq](#)
- struct [has\\_plus\\_impl](#)< C >
- struct [has\\_mul\\_impl](#)< C >
- struct [has\\_mul\\_eq\\_impl](#)< C >
- struct [has\\_plus\\_eq\\_impl](#)< C >
- struct [has\\_minus\\_eq\\_impl](#)< C >
- struct [has\\_minus\\_impl](#)< C >
- struct [has\\_operation](#)< T >

### Namespaces

- namespace [FFLAS](#)

### Typedefs

- using [ZOSparseMatrix](#) = std::true\_type
- using [NotZOSparseMatrix](#) = std::false\_type
- using [SimdSparseMatrix](#) = std::true\_type
- using [NoSimdSparseMatrix](#) = std::false\_type
- using [MKLSparseMatrixFormat](#) = std::true\_type
- using [NotMKLSparseMatrixFormat](#) = std::false\_type
- template<class T >  
using [has\\_plus](#) = typename std::conditional<std::is\_arithmetic<T>::value, std::true\_type, [has\\_plus\\_impl](#)<T>>::type

- `template<class T >`  
using `has_minus` = `typename std::conditional<std::is_arithmetic<T>::value, std::true_type, has_minus_impl<T>>::type`
- `template<class T >`  
using `has_equal` = `typename std::conditional<std::is_arithmetic<T>::value, std::true_type, std::is_copy_assignable<T>>::type`
- `template<class T >`  
using `has_plus_eq` = `typename std::conditional<std::is_arithmetic<T>::value, std::true_type, has_plus_eq_impl<T>>::type`
- `template<class T >`  
using `has_minus_eq` = `typename std::conditional<std::is_arithmetic<T>::value, std::true_type, has_minus_eq_impl<T>>::type`
- `template<class T >`  
using `has_mul` = `typename std::conditional<std::is_arithmetic<T>::value, std::true_type, has_mul_impl<T>>::type`
- `template<class T >`  
using `has_mul_eq` = `typename std::conditional<std::is_arithmetic<T>::value, std::true_type, has_mul_eq_impl<T>>::type`

## 13.177 utils.h File Reference

```
#include <algorithm>
#include <numeric>
#include <vector>
```

### Data Structures

- struct [StatsMatrix](#)

### Namespaces

- namespace [FFLAS](#)

### Functions

- `template<class It >`  
double [computeDeviation](#) (It begin, It end)
- `template<class Field >`  
[StatsMatrix](#) [getStat](#) (const [Field](#) &F, const [index\\_t](#) \*row, const [index\\_t](#) \*col, `typename Field::ConstElement_ptr` val, [uint64\\_t](#) rowdim, [uint64\\_t](#) coldim, [uint64\\_t](#) nnz)

## 13.178 fflas\_transpose.h File Reference

transpose the storage of the matrix (switch between row and col major mode)

```
#include "fflas-ffpack/utils/debug.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/fflas_memory.h"
#include "fflas-ffpack/fflas/fflas_simd.h"
```

### Data Structures

- struct [BlockTransposeSIMD](#)< [Field](#), [Simd](#), >

## Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::\\_ftranspose\\_impl](#)

## Macros

- `#define FFLAS_TRANSPOSE_BLOCKSIZE 32`
- `#define LD(i)`
- `#define ST(i)`

## Functions

- `template<size_t bs, typename Field, typename BTSimd >`  
`void not_inplace (const Field &F, const BTSimd &BTS, const size_t m, const size_t n, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb)`
- `template<size_t bs, typename Field, typename BTSimd >`  
`void square_inplace (const Field &F, const BTSimd &BTS, const size_t m, typename Field::Element_ptr A, const size_t lda)`
- `template<size_t bs, typename Field, typename BTSimd >`  
`void nonsquare_inplace_v1 (const Field &F, const BTSimd &BTS, const size_t m, const size_t n, typename Field::Element_ptr A)`
- `template<size_t bs, typename Field, typename BTSimd >`  
`void nonsquare_inplace_v2 (const Field &F, const BTSimd &BTS, const size_t m, const size_t n, typename Field::Element_ptr A)`

### 13.178.1 Detailed Description

transpose the storage of the matrix (switch between row and col major mode)

### 13.178.2 Macro Definition Documentation

#### 13.178.2.1 FFLAS\_TRANSPOSE\_BLOCKSIZE

```
#define FFLAS_TRANSPOSE_BLOCKSIZE 32
```

#### 13.178.2.2 LD

```
#define LD(  
    i)
```

##### Value:

```
R##i=Simd::loadu(A+lda*i)
```

#### 13.178.2.3 ST

```
#define ST(  
    i)
```

##### Value:

```
Simd::storeu(B+ldb*i,R##i)
```

## 13.179 ffpack.dox File Reference

## 13.180 ffpack.h File Reference

Set of elimination based routines for dense linear algebra.

```
#include "givaro/givpoly1.h"  
#include <fflas-ffpack/fflas-ffpack-config.h>  
#include "fflas-ffpack/fflas/fflas.h"  
#include "fflas-ffpack/fflas/fflas_helpers.inl"
```



```

#include <list>
#include <vector>
#include <iostream>
#include <algorithm>
#include "fflas-ffpack/checkers/checkers_ffpack.h"
#include "ffpack_fgesv.inl"
#include "ffpack_fgetrs.inl"
#include "fflas-ffpack/checkers/checkers_ffpack.inl"
#include "ffpack_pluq.inl"
#include "ffpack_pluq_mp.inl"
#include "ffpack_ppluq.inl"
#include "ffpack_ludivine.inl"
#include "ffpack_ludivine_mp.inl"
#include "ffpack_echelonforms.inl"
#include "ffpack_fsytrf.inl"
#include "ffpack_invert.inl"
#include "ffpack_ftrtr.inl"
#include "ffpack_ftrstr.inl"
#include "ffpack_ftrssyr2k.inl"
#include "ffpack_charpoly_kglu.inl"
#include "ffpack_charpoly_kgfast.inl"
#include "ffpack_charpoly_kgfastgeneralized.inl"
#include "ffpack_charpoly_danilevski.inl"
#include "ffpack_charpoly.inl"
#include "ffpack_frobenius.inl"
#include "ffpack_minpoly.inl"
#include "ffpack_krylovelim.inl"
#include "ffpack_permutation.inl"
#include "ffpack_rankprofiles.inl"
#include "ffpack_det_mp.inl"
#include "ffpack_bruhatgen.inl"
#include "ffpack.inl"

```

## Data Structures

- class [CharpolyFailed](#)

## Namespaces

- namespace [FFPACK](#)  
*Finite Field PACK Set of elimination based routines for dense linear algebra.*
- namespace [FFPACK::Protected](#)

## Macros

- `#define` [\\_\\_FFLASFFPACK\\_FTRSTR\\_THRESHOLD](#) 64
- `#define` [\\_\\_FFLASFFPACK\\_FTRSSYR2K\\_THRESHOLD](#) 64

## Functions

- void [LAPACKPerm2MathPerm](#) (size\_t \*MathP, const size\_t \*LapackP, const size\_t N)  
*Conversion of a permutation from LAPACK format to Math format.*
- void [MathPerm2LAPACKPerm](#) (size\_t \*LapackP, const size\_t \*MathP, const size\_t N)  
*Conversion of a permutation from Maths format to LAPACK format.*
- template<class [Field](#) >  
void [applyP](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const [FFLAS::FFLAS\\_TRANSPOSE](#) Trans,

const size\_t M, const size\_t ibeg, const size\_t iend, typename Field::Element\_ptr A, const size\_t lda, const size\_t \*P)

*Computes  $P1 \times Diag(I_R, P2)$  where  $P1$  is a LAPACK and  $P2$  a LAPACK permutation and store the result in  $P1$  as a LAPACK permutation.*

- template<class Field >  
void applyP (const Field &F, const FFLAS::FFLAS\_SIDE Side, const FFLAS::FFLAS\_TRANSPOSE Trans, const size\_t m, const size\_t ibeg, const size\_t iend, typename Field::Element\_ptr A, const size\_t lda, const size\_t \*P, const FFLAS::ParSeqHelper::Sequential seq)
  - template<class Field , class Cut , class Param >  
void applyP (const Field &F, const FFLAS::FFLAS\_SIDE Side, const FFLAS::FFLAS\_TRANSPOSE Trans, const size\_t m, const size\_t ibeg, const size\_t iend, typename Field::Element\_ptr A, const size\_t lda, const size\_t \*P, const FFLAS::ParSeqHelper::Parallel< Cut, Param > par)
  - template<class Field >  
void MonotonicApplyP (const Field &F, const FFLAS::FFLAS\_SIDE Side, const FFLAS::FFLAS\_TRANSPOSE Trans, const size\_t M, const size\_t ibeg, const size\_t iend, typename Field::Element\_ptr A, const size\_t lda, const size\_t \*P, const size\_t R)
- Apply a R-monotonically increasing permutation P, to the matrix A.*
- template<class Field >  
void fgetrs (const Field &F, const FFLAS::FFLAS\_SIDE Side, const size\_t M, const size\_t N, const size\_t R, typename Field::Element\_ptr A, const size\_t lda, const size\_t \*P, const size\_t \*Q, typename Field::Element\_ptr B, const size\_t ldb, int \*info)
- Solve the system  $AX = B$  or  $XA = B$ .*
- template<class Field >  
Field::Element\_ptr fgetrs (const Field &F, const FFLAS::FFLAS\_SIDE Side, const size\_t M, const size\_t N, const size\_t NRHS, const size\_t R, typename Field::Element\_ptr A, const size\_t lda, const size\_t \*P, const size\_t \*Q, typename Field::Element\_ptr X, const size\_t ldx, typename Field::ConstElement\_ptr B, const size\_t ldb, int \*info)
- Solve the system  $A X = B$  or  $X A = B$ .*
- template<class Field >  
size\_t fgesv (const Field &F, const FFLAS::FFLAS\_SIDE Side, const size\_t M, const size\_t N, typename Field::Element\_ptr A, const size\_t lda, typename Field::Element\_ptr B, const size\_t ldb, int \*info)
- Square system solver.*
- template<class Field >  
size\_t fgesv (const Field &F, const FFLAS::FFLAS\_SIDE Side, const size\_t M, const size\_t N, const size\_t NRHS, typename Field::Element\_ptr A, const size\_t lda, typename Field::Element\_ptr X, const size\_t ldx, typename Field::ConstElement\_ptr B, const size\_t ldb, int \*info)
- Rectangular system solver.*
- template<class Field >  
void ftrtri (const Field &F, const FFLAS::FFLAS\_UPLO Uplo, const FFLAS::FFLAS\_DIAG Diag, const size\_t N, typename Field::Element\_ptr A, const size\_t lda, const size\_t threshold=\_\_FFLASFFPACK\_FTRTRI\_THRESHOLD)
- Compute the inverse of a triangular matrix.*
- template<class Field >  
void trinv\_left (const Field &F, const size\_t N, typename Field::ConstElement\_ptr L, const size\_t ldl, typename Field::Element\_ptr X, const size\_t ldx)
  - template<class Field >  
void ftrtrm (const Field &F, const FFLAS::FFLAS\_SIDE side, const FFLAS::FFLAS\_DIAG diag, const size\_t N, typename Field::Element\_ptr A, const size\_t lda)
- Compute the product of two triangular matrices of opposite shape.*
- template<class Field >  
void ftrstr (const Field &F, const FFLAS::FFLAS\_SIDE side, const FFLAS::FFLAS\_UPLO Uplo, const FFLAS::FFLAS\_DIAG diagA, const FFLAS::FFLAS\_DIAG diagB, const size\_t N, typename Field::ConstElement\_ptr A, const size\_t lda, typename Field::Element\_ptr B, const size\_t ldb, const size\_t threshold=\_\_FFLASFFPACK\_FTRSTR\_THRESHOLD)
- Solve a triangular system with a triangular right hand side of the same shape.*

- `template<class Field >`  
`void ftrssyr2k (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diagA, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, const size_t threshold=__FFLASFFPACK_FTRSSYR2K_THRESHOLD)`  
*Solve a triangular system in a symmetric sum: find B upper/lower triangular such that  $A^T B + B^T A = C$  where C is symmetric.*
- `template<class Field >`  
`bool fsytrf (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, const size_t threshold=__FFLASFFPACK_FSYTRF_THRESHOLD)`  
*Triangular factorization of symmetric matrices.*
- `template<class Field >`  
`bool fsytrf (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, const FFLAS::ParSeqHelper::Sequential seq, const size_t threshold=__FFLASFFPACK_FSYTRF_THRESHOLD)`
- `template<class Field , class Cut , class Param >`  
`bool fsytrf (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, const FFLAS::ParSeqHelper::Parallel< Cut, Param > par, const size_t threshold=__FFLASFFPACK_FSYTRF_THRESHOLD)`
- `template<class Field >`  
`bool fsytrf_nonunit (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr D, const size_t incD, const size_t threshold=__FFLASFFPACK_FSYTRF_THRESHOLD)`  
*Triangular factorization of symmetric matrices.*
- `template<class Field >`  
`size_t PLUQ (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q)`  
*Compute a PLUQ factorization of the given matrix.*
- `template<class Field >`  
`size_t pPLUQ (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q)`
- `template<class Field >`  
`size_t PLUQ (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, const FFLAS::ParSeqHelper::Sequential &PHelper, size_t BCThreshold=__FFLASFFPACK_PLUQ_THRESHOLD)`
- `template<class Field , class Cut , class Param >`  
`size_t PLUQ (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, const FFLAS::ParSeqHelper::Parallel< Cut, Param > &PHelper)`
- `template<class Field >`  
`size_t LUdivine (const Field &F, const FFLAS::FFLAS_DIAG Diag, const FFLAS::FFLAS_TRANSPOSE trans, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const FFPACK_LU_TAG LuTag=FpackSlabRecursive, const size_t cutoff=__FFLASFFPACK_LUDIVINE_THRESHOLD)`  
*Compute the CUP or PLE factorization of the given matrix.*
- `template<class Field >`  
`size_t LUdivine_construct (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr X, const size_t idx, typename Field::Element_ptr u, const size_t incu, size_t *P, bool computeX, const FFPACK_MINPOLY_TAG MinTag=FpackDense, const size_t kg_mc=0, const size_t kg_mb=0, const size_t kg_j=0)`
- `template<class Field >`  
`size_t ColumnEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, bool transform=false, const FFPACK_LU_TAG LuTag=FpackSlabRecursive)`  
*Compute the Column Echelon form of the input matrix in-place.*
- `template<class Field >`  
`size_t pColumnEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, bool transform=false, size_t numthreads=0, const FFPACK_LU_TAG LuTag=FpackTileRecursive)`

- `template<class Field , class PSHelper >`  
`size_t ColumnEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A,`  
`const size_t lda, size_t *P, size_t *Qt, const bool transform, const FFPACK_LU_TAG LuTag, const PSHelper`  
`&psH)`
- `template<class Field >`  
`size_t RowEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr`  
`A, const size_t lda, size_t *P, size_t *Qt, const bool transform=false, const FFPACK_LU_TAG LuTag=`  
`FfpackSlabRecursive)`  
*Compute the Row Echelon form of the input matrix in-place.*
- `template<class Field >`  
`size_t pRowEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A,`  
`const size_t lda, size_t *P, size_t *Qt, const bool transform=false, size_t numthreads=0, const FFPACK_`  
`LU_TAG LuTag=FfpackTileRecursive)`
- `template<class Field , class PSHelper >`  
`size_t RowEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A,`  
`const size_t lda, size_t *P, size_t *Qt, const bool transform, const FFPACK_LU_TAG LuTag, const PSHelper`  
`&psH)`
- `template<class Field >`  
`size_t ReducedColumnEchelonForm (const Field &F, const size_t M, const size_t N, typename`  
`Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform=false, const FFPACK_`  
`LU_TAG LuTag=FfpackSlabRecursive)`  
*Compute the Reduced Column Echelon form of the input matrix in-place.*
- `template<class Field >`  
`size_t pReducedColumnEchelonForm (const Field &F, const size_t M, const size_t N, typename`  
`Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform=false, size_t numthreads=0,`  
`const FFPACK_LU_TAG LuTag=FfpackTileRecursive)`
- `template<class Field , class PSHelper >`  
`size_t ReducedColumnEchelonForm (const Field &F, const size_t M, const size_t N, typename`  
`Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform, const FFPACK_LU_`  
`_TAG LuTag, const PSHelper &psH)`
- `template<class Field >`  
`size_t ReducedRowEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr`  
`A, const size_t lda, size_t *P, size_t *Qt, const bool transform=false, const FFPACK_LU_TAG LuTag=`  
`FfpackSlabRecursive)`  
*Compute the Reduced Row Echelon form of the input matrix in-place.*
- `template<class Field >`  
`size_t pReducedRowEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr`  
`A, const size_t lda, size_t *P, size_t *Qt, const bool transform=false, size_t numthreads=0, const FFPACK_`  
`_LU_TAG LuTag=FfpackTileRecursive)`
- `template<class Field , class PSHelper >`  
`size_t ReducedRowEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr`  
`A, const size_t lda, size_t *P, size_t *Qt, const bool transform, const FFPACK_LU_TAG LuTag, const`  
`PSHelper &psH)`
- `template<class Field >`  
`size_t GaussJordan (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const`  
`size_t lda, const size_t colbeg, const size_t rowbeg, const size_t colsize, size_t *P, size_t *Q, const`  
`FFPACK::FFPACK_LU_TAG LuTag)`  
*Gauss-Jordan algorithm computing the Reduced Row echelon form and its transform matrix.*
- `template<class Field >`  
`Field::Element_ptr Invert (const Field &F, const size_t M, typename Field::Element_ptr A, const size_t lda, int`  
`&nullity)`  
*Invert the given matrix in place or computes its nullity if it is singular.*
- `template<class Field >`  
`Field::Element_ptr Invert (const Field &F, const size_t M, typename Field::ConstElement_ptr A, const size_t`  
`lda, typename Field::Element_ptr X, const size_t ldx, int &nullity)`  
*Invert the given matrix or computes its nullity if it is singular.*

- `template<class Field >`  
`Field::Element_ptr Invert2` (const `Field` &F, const `size_t` M, typename `Field::Element_ptr` A, const `size_t` Ida, typename `Field::Element_ptr` X, const `size_t` Idx, int &nullity)  
*Invert the given matrix or computes its nullity if it is singular.*
- `template<class PolRing >`  
`std::list< typename PolRing::Element > & CharPoly` (const `PolRing` &R, `std::list< typename PolRing::Element > &charp`, const `size_t` N, typename `PolRing::Domain_t::Element_ptr` A, const `size_t` Ida, typename `PolRing::Domain_t::Randlter` &G, const `FFPACK_CHARPOLY_TAG` CharpTag=`FfpackAuto`, const `size_t` degree=`__FFLASFFPACK_ARITHPROG_THRESHOLD`)  
*Compute the characteristic polynomial of the matrix A.*
- `template<class PolRing >`  
`PolRing::Element & CharPoly` (const `PolRing` &R, typename `PolRing::Element` &charp, const `size_t` N, typename `PolRing::Domain_t::Element_ptr` A, const `size_t` Ida, typename `PolRing::Domain_t::Randlter` &G, const `FFPACK_CHARPOLY_TAG` CharpTag=`FfpackAuto`, const `size_t` degree=`__FFLASFFPACK_ARITHPROG_THRESHOLD`)  
*Compute the characteristic polynomial of the matrix A.*
- `template<class PolRing >`  
`PolRing::Element & CharPoly` (const `PolRing` &R, typename `PolRing::Element` &charp, const `size_t` N, typename `PolRing::Domain_t::Element_ptr` A, const `size_t` Ida, const `FFPACK_CHARPOLY_TAG` CharpTag=`FfpackAuto`, const `size_t` degree=`__FFLASFFPACK_ARITHPROG_THRESHOLD`)  
*Compute the characteristic polynomial of the matrix A.*
- `template<class Field , class Polynomial >`  
`std::list< Polynomial > & KellerGehrig` (const `Field` &F, `std::list< Polynomial > &charp`, const `size_t` N, typename `Field::ConstElement_ptr` A, const `size_t` Ida)
- `template<class Field , class Polynomial >`  
`int KGFast` (const `Field` &F, `std::list< Polynomial > &charp`, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` Ida, `size_t` \*kg\_mc, `size_t` \*kg\_mb, `size_t` \*kg\_j)
- `template<class Field , class Polynomial >`  
`std::list< Polynomial > & KGFast_generalized` (const `Field` &F, `std::list< Polynomial > &charp`, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` Ida)
- `template<class Field >`  
`void fgemv_kgf` (const `Field` &F, const `size_t` N, typename `Field::ConstElement_ptr` A, const `size_t` Ida, typename `Field::ConstElement_ptr` X, const `size_t` incX, typename `Field::Element_ptr` Y, const `size_t` incY, const `size_t` kg\_mc, const `size_t` kg\_mb, const `size_t` kg\_j)
- `template<class Field , class Polynomial , class Randlter >`  
`std::list< Polynomial > & LUKrylov` (const `Field` &F, `std::list< Polynomial > &charp`, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` Ida, typename `Field::Element_ptr` U, const `size_t` ldu, `Randlter` &G)
- `template<class Field , class Polynomial >`  
`std::list< Polynomial > & Danilevski` (const `Field` &F, `std::list< Polynomial > &charp`, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` Ida)
- `template<class PolRing >`  
`void RandomKrylovPrecond` (const `PolRing` &PR, `std::list< typename PolRing::Element > &completedFactors`, const `size_t` N, typename `PolRing::Domain_t::Element_ptr` A, const `size_t` Ida, `size_t` &Nb, typename `PolRing::Domain_t::Element_ptr` &B, `size_t` &ldb, typename `PolRing::Domain_t::Randlter` &g, const `size_t` degree=`__FFLASFFPACK_ARITHPROG_THRESHOLD`)
- `template<class PolRing >`  
`std::list< typename PolRing::Element > & ArithProg` (const `PolRing` &PR, `std::list< typename PolRing::Element > &frobeniusForm`, const `size_t` N, typename `PolRing::Domain_t::Element_ptr` A, const `size_t` Ida, const `size_t` degree)
- `template<class Field , class Polynomial >`  
`std::list< Polynomial > & LUKrylov_KGFast` (const `Field` &F, `std::list< Polynomial > &charp`, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` Ida, typename `Field::Element_ptr` X, const `size_t` Idx)
- `template<class Field , class Polynomial >`  
`Polynomial & MinPoly` (const `Field` &F, `Polynomial` &minP, const `size_t` N, typename `Field::ConstElement_ptr` A, const `size_t` Ida)  
*Compute the minimal polynomial of the matrix A.*

- template<class [Field](#) , class Polynomial , class RandIter >  
Polynomial & [MinPoly](#) (const [Field](#) &F, Polynomial &minP, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, RandIter &G)  
*Compute the minimal polynomial of the matrix A.*
- template<class [Field](#) , class Polynomial >  
Polynomial & [MatVecMinPoly](#) (const [Field](#) &F, Polynomial &minP, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) v, const size\_t incv)  
*Compute the minimal polynomial of the matrix A and a vector v, namely the first linear dependency relation in the Krylov basis  $(v, Av, \dots, A^N v)$ .*
- template<class [Field](#) , class Polynomial >  
Polynomial & [MatVecMinPoly](#) (const [Field](#) &F, Polynomial &minP, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) v, const size\_t incv, typename [Field::Element\\_ptr](#) K, const size\_t ldk, size\_t \*P)  
*Compute the minimal polynomial of the matrix A and a vector v, namely the first linear dependency relation in the Krylov basis  $(v, Av, \dots, A^N v)$ .*
- template<class [Field](#) , class Polynomial >  
Polynomial & [Hybrid\\_KGF\\_LUK\\_MinPoly](#) (const [Field](#) &F, Polynomial &minP, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) X, const size\_t ldx, size\_t \*P, const FFPACK\_MINPOLY\_TAG MinTag=FFPACK::FfpackDense, const size\_t kg\_mc=0, const size\_t kg\_↔ mb=0, const size\_t kg\_j=0)
- template<class [Field](#) >  
size\_t [Rank](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*Computes the rank of the given matrix using a PLUQ factorization.*
- template<class [Field](#) >  
size\_t [pRank](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t numthreads=0)
- template<class [Field](#) , class PSHelper >  
size\_t [Rank](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, const PSHelper &psH)
- template<class [Field](#) >  
bool [IsSingular](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*Returns true if the given matrix is singular.*
- template<class [Field](#) >  
[Field::Element](#) & [Det](#) (const [Field](#) &F, typename [Field::Element](#) &det, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*P=NULL, size\_t \*Q=NULL)  
*Returns the determinant of the given square matrix.*
- template<class [Field](#) >  
[Field::Element](#) & [pDet](#) (const [Field](#) &F, typename [Field::Element](#) &det, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t numthreads=0, size\_t \*P=NULL, size\_t \*Q=NULL)
- template<class [Field](#) , class PSHelper >  
[Field::Element](#) & [Det](#) (const [Field](#) &F, typename [Field::Element](#) &det, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, const PSHelper &psH, size\_t \*P=NULL, size\_t \*Q=NULL)
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [Solve](#) (const [Field](#) &F, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) x, const int incx, typename [Field::ConstElement\\_ptr](#) b, const int incb)  
*Solves a linear system  $AX = b$  using PLUQ factorization.*
- template<class [Field](#) , class PSHelper >  
[Field::Element\\_ptr](#) [Solve](#) (const [Field](#) &F, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) x, const int incx, typename [Field::ConstElement\\_ptr](#) b, const int incb, PSHelper &psH)
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [pSolve](#) (const [Field](#) &F, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) x, const int incx, typename [Field::ConstElement\\_ptr](#) b, const int incb, size\_t numthreads=0)
- template<class [Field](#) >  
\*void [RandomNullSpaceVector](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) X, const size\_t incX)



*Solve  $LX = B$  or  $XL = B$  in place.*

- template<class [Field](#) >  
size\_t [NullSpaceBasis](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) &NS, size\_t &ldn, size\_t &NSdim)

*Computes a basis of the Left/Right nullspace of the matrix A.*

- template<class [Field](#) >  
size\_t [RowRankProfile](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*&rkprofile, const FFPACK\_LU\_TAG LuTag=[FfpackSlabRecursive](#))

*Computes the row rank profile of A.*

- template<class [Field](#) >  
size\_t [pRowRankProfile](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*&rkprofile, size\_t numthreads=0, const FFPACK\_LU\_TAG LuTag=[FfpackTileRecursive](#))
- template<class [Field](#), class PSHelper >  
size\_t [RowRankProfile](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*&rkprofile, const FFPACK\_LU\_TAG LuTag, PSHelper &psH)
- template<class [Field](#) >  
size\_t [ColumnRankProfile](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*&rkprofile, const FFPACK\_LU\_TAG LuTag=[FfpackSlabRecursive](#))

*Computes the column rank profile of A.*

- template<class [Field](#) >  
size\_t [pColumnRankProfile](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*&rkprofile, size\_t numthreads=0, const FFPACK\_LU\_TAG LuTag=[FfpackTileRecursive](#))
- template<class [Field](#), class PSHelper >  
size\_t [ColumnRankProfile](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*&rkprofile, const FFPACK\_LU\_TAG LuTag, PSHelper &psH)
- void [RankProfileFromLU](#) (const size\_t \*P, const size\_t N, const size\_t R, size\_t \*rkprofile, const FFPACK\_LU\_TAG LuTag)

*Recovers the column/row rank profile from the permutation of an LU decomposition.*

- size\_t [LeadingSubmatrixRankProfiles](#) (const size\_t M, const size\_t N, const size\_t R, const size\_t LSm, const size\_t LSn, const size\_t \*P, const size\_t \*Q, size\_t \*RRP, size\_t \*CRP)

*Recovers the row and column rank profiles of any leading submatrix from the PLUQ decomposition.*

- template<class [Field](#) >  
size\_t [RowRankProfileSubmatrixIndices](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*&rowindices, size\_t \*&colindices, size\_t &R)

*RowRankProfileSubmatrixIndices.*

- template<class [Field](#) >  
size\_t [ColRankProfileSubmatrixIndices](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*&rowindices, size\_t \*&colindices, size\_t &R)

*Computes the indices of the submatrix  $r \times r$  X of A whose columns correspond to the column rank profile of A.*

- template<class [Field](#) >  
size\_t [RowRankProfileSubmatrix](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) &X, size\_t &R)

*Computes the  $r \times r$  submatrix X of A, by picking the row rank profile rows of A.*

- template<class [Field](#) >  
size\_t [ColRankProfileSubmatrix](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) &X, size\_t &R)

*Compute the  $r \times r$  submatrix X of A, by picking the row rank profile rows of A.*

- template<class [Field](#) >  
void [getTriangular](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const [FFLAS::FFLAS\\_DIAG](#) diag, const size\_t M, const size\_t N, const size\_t R, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) T, const size\_t ldt, const bool OnlyNonZeroVectors=false)

*Extracts a triangular matrix from a compact storage  $A=L|U$  of rank R.*

- template<class [Field](#) >  
void [getTriangular](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const [FFLAS::FFLAS\\_DIAG](#) diag, const size\_t M, const size\_t N, const size\_t R, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*Cleans up a compact storage  $A=L\backslash U$  to reveal a triangular matrix of rank R.*
- template<class [Field](#) >  
void [getEchelonForm](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const [FFLAS::FFLAS\\_DIAG](#) diag, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) T, const size\_t ldt, const bool OnlyNonZeroVectors=false, const [FFPACK\\_LU\\_TAG](#) LuTag=[FpackSlabRecursive](#))  
*Extracts a matrix in echelon form from a compact storage  $A=L\backslash U$  of rank R obtained by RowEchelonForm or ColumnEchelonForm.*
- template<class [Field](#) >  
void [getEchelonForm](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const [FFLAS::FFLAS\\_DIAG](#) diag, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, typename [Field::Element\\_ptr](#) A, const size\_t lda, const [FFPACK\\_LU\\_TAG](#) LuTag=[FpackSlabRecursive](#))  
*Cleans up a compact storage  $A=L\backslash U$  obtained by RowEchelonForm or ColumnEchelonForm to reveal an echelon form of rank R.*
- template<class [Field](#) >  
void [getEchelonTransform](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const [FFLAS::FFLAS\\_DIAG](#) diag, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const size\_t \*Q, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) T, const size\_t ldt, const [FFPACK\\_LU\\_TAG](#) LuTag=[FpackSlabRecursive](#))  
*Extracts a transformation matrix to echelon form from a compact storage  $A=L\backslash U$  of rank R obtained by RowEchelonForm or ColumnEchelonForm.*
- template<class [Field](#) >  
void [getReducedEchelonForm](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) T, const size\_t ldt, const bool OnlyNonZeroVectors=false, const [FFPACK\\_LU\\_TAG](#) LuTag=[FpackSlabRecursive](#))  
*Extracts a matrix in echelon form from a compact storage  $A=L\backslash U$  of rank R obtained by ReducedRowEchelonForm or ReducedColumnEchelonForm with transform = true.*
- template<class [Field](#) >  
void [getReducedEchelonForm](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, typename [Field::Element\\_ptr](#) A, const size\_t lda, const [FFPACK\\_LU\\_TAG](#) LuTag=[FpackSlabRecursive](#))  
*Cleans up a compact storage  $A=L\backslash U$  of rank R obtained by ReducedRowEchelonForm or ReducedColumnEchelonForm with transform = true.*
- template<class [Field](#) >  
void [getReducedEchelonTransform](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const size\_t \*Q, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) T, const size\_t ldt, const [FFPACK\\_LU\\_TAG](#) LuTag=[FpackSlabRecursive](#))  
*Extracts a transformation matrix to echelon form from a compact storage  $A=L\backslash U$  of rank R obtained by RowEchelonForm or ColumnEchelonForm.*
- void [PLUQtoEchelonPermutation](#) (const size\_t N, const size\_t R, const size\_t \*P, size\_t \*outPerm)  
*Auxiliary routine: determines the permutation that changes a PLUQ decomposition into a echelon form revealing PLUQ decomposition.*
- template<class [Field](#) >  
size\_t [LTBruhatGen](#) (const [Field](#) &Fi, const [FFLAS::FFLAS\\_DIAG](#) diag, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*P, size\_t \*Q)  
*LTBruhatGen Suppose A is Left Triangular Matrix This procedure computes the Bruhat Representation of A and return the rank of A.*
- template<class [Field](#) >  
void [getLTBruhatGen](#) (const [Field](#) &Fi, const size\_t N, const size\_t r, const size\_t \*P, const size\_t \*Q, typename [Field::Element\\_ptr](#) R, const size\_t ldr)  
*GetLTBruhatGen This procedure Computes the Rank Revealing Matrix based on the Bruhta representation of a Matrix.*



- `template<class Field >`  
`void getLTBruhatGen (const Field &Fi, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag, const size_t N, const size_t r, const size_t *P, const size_t *Q, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr T, const size_t ldt)`  
*GetLTBruhatGen This procedure computes the matrix L or U of the Bruhat Representation Suppose that A is the bruhat representation of a matrix.*
- `size_t LTQSorder (const size_t N, const size_t r, const size_t *P, const size_t *Q)`  
*LTQSorder This procedure computes the order of quasiseparability of a matrix.*
- `template<class Field >`  
`size_t CompressToBlockBiDiagonal (const Field &Fi, const FFLAS::FFLAS_UPLO Uplo, size_t N, size_t s, size_t r, const size_t *P, const size_t *Q, typename Field::Element_ptr A, size_t lda, typename Field::Element_ptr X, size_t ldx, size_t *K, size_t *M, size_t *T)`  
*CompressToBlockBiDiagonal This procedure compress a compact representation of a row echelon form or column echelon form.*
- `template<class Field >`  
`void ExpandBlockBiDiagonalToBruhat (const Field &Fi, const FFLAS::FFLAS_UPLO Uplo, size_t N, size_t s, size_t r, typename Field::Element_ptr A, size_t lda, typename Field::Element_ptr X, size_t ldx, size_t NbBlocks, size_t *K, size_t *M, size_t *T)`  
*ExpandBlockBiDiagonal This procedure expand a compact representation of a row echelon form or column echelon form.*
- `void Bruhat2EchelonPermutation (size_t N, size_t R, const size_t *P, const size_t *Q, size_t *M)`  
*Bruhat2EchelonPermutation (N,R,P,Q) Compute M such that LM or MU is in echelon form where L or U are factors of the Bruhat Representation.*
- `size_t * Tinverter (size_t *T, size_t r)`
- `template<class Field >`  
`void ComputeRPermutation (const Field &Fi, size_t N, size_t r, const size_t *P, const size_t *Q, size_t *R, size_t *MU, size_t *ML)`
- `template<class Field >`  
`void productBruhatxTS (const Field &Fi, size_t N, size_t s, size_t r, const size_t *P, const size_t *Q, const typename Field::Element_ptr Xu, size_t ldu, size_t NbBlocksU, size_t *Ku, size_t *Tu, size_t *MU, const typename Field::Element_ptr XI, size_t ldl, size_t NbBlocksL, size_t *KI, size_t *TI, size_t *ML, typename Field::Element_ptr B, size_t t, size_t ldb, typename Field::Element_ptr C, size_t ldc)`  
*productBruhatxTS Compute the product between the CRE compact representation of a matrix A and B a tall matrix*
- `template<class Field >`  
`Field::Element_ptr LQUPtoInverseOfFullRankMinor (const Field &F, const size_t rank, typename Field::Element_ptr A_factors, const size_t lda, const size_t *QtPointer, typename Field::Element_ptr X, const size_t ldx)`  
*LQUPtoInverseOfFullRankMinor.*

## 13.180.1 Detailed Description

Set of elimination based routines for dense linear algebra.

Matrices are supposed over finite prime field of characteristic less than  $2^{26}$ .

## 13.180.2 Macro Definition Documentation

### 13.180.2.1 \_\_FFLASFFPACK\_FTRSTR\_THRESHOLD

```
#define __FFLASFFPACK_FTRSTR_THRESHOLD 64
```

### 13.180.2.2 \_\_FFLASFFPACK\_FTRSSYR2K\_THRESHOLD

```
#define __FFLASFFPACK_FTRSSYR2K_THRESHOLD 64
```

## 13.181 ffpack.inl File Reference

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [\\_\\_FFLASFFPACK\\_ffpack\\_INL](#)

### Functions

- template<class [Field](#) >  
size\_t [Rank](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t Ida)  
*Computes the rank of the given matrix using a PLUQ factorization.*
- template<class [Field](#) >  
size\_t [pRank](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t Ida, size\_t numthreads=0)
- template<class [Field](#), class PSHelper >  
size\_t [Rank](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t Ida, const PSHelper &psH)
- template<class [Field](#) >  
bool [IsSingular](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t Ida)  
*Returns true if the given matrix is singular.*
- template<class [Field](#) >  
[Field::Element](#) & [Det](#) (const [Field](#) &F, typename [Field::Element](#) &det, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t Ida, size\_t \*P=NULL, size\_t \*Q=NULL)  
*Returns the determinant of the given square matrix.*
- template<class [Field](#) >  
[Field::Element](#) & [pDet](#) (const [Field](#) &F, typename [Field::Element](#) &det, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t Ida, size\_t numthreads=0, size\_t \*P=NULL, size\_t \*Q=NULL)
- template<class [Field](#), class PSHelper >  
[Field::Element](#) & [Det](#) (const [Field](#) &F, typename [Field::Element](#) &det, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t Ida, const PSHelper &psH, size\_t \*P=NULL, size\_t \*Q=NULL)
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [Solve](#) (const [Field](#) &F, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t Ida, typename [Field::Element\\_ptr](#) x, const int incx, typename [Field::ConstElement\\_ptr](#) b, const int incb)  
*Solves a linear system  $AX = b$  using PLUQ factorization.*
- template<class [Field](#), class PSHelper >  
[Field::Element\\_ptr](#) [Solve](#) (const [Field](#) &F, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t Ida, typename [Field::Element\\_ptr](#) x, const int incx, typename [Field::ConstElement\\_ptr](#) b, const int incb, PSHelper &psH)
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [pSolve](#) (const [Field](#) &F, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t Ida, typename [Field::Element\\_ptr](#) x, const int incx, typename [Field::ConstElement\\_ptr](#) b, const int incb, size\_t numthreads=0)
- template<class [Field](#) >  
void [RandomNullSpaceVector](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t Ida, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
*Solve  $LX = B$  or  $XL = B$  in place.*
- template<class [Field](#) >  
size\_t [NullSpaceBasis](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t Ida, typename [Field::Element\\_ptr](#) &NS, size\_t &Idn, size\_t &NSdim)  
*Computes a basis of the Left/Right nullspace of the matrix A.*

- template<class [Field](#) >  
void [solveLB](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t R, typename [Field::Element\\_ptr](#) L, const size\_t ldl, const size\_t \*Q, typename [Field::Element\\_ptr](#) B, const size\_t ldb)
- template<class [Field](#) >  
void [solveLB2](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t R, typename [Field::Element\\_ptr](#) L, const size\_t ldl, const size\_t \*Q, typename [Field::Element\\_ptr](#) B, const size\_t ldb)

### 13.181.1 Macro Definition Documentation

#### 13.181.1.1 \_\_FFLASFFPACK\_ffpack\_INL

```
#define __FFLASFFPACK_ffpack_INL
```

## 13.182 ffpack\_bruhatgen.inl File Reference

### Namespaces

- namespace [FFPACK](#)  
*Finite **Field** **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [\\_\\_FFLASFFPACK\\_ffpack\\_bruhatgen\\_inl](#)

### Functions

- template<class [Field](#) >  
size\_t [LTBruhatGen](#) (const [Field](#) &Fi, const [FFLAS::FFLAS\\_DIAG](#) diag, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*P, size\_t \*Q)  
*LTBruhatGen Suppose A is Left Triangular Matrix This procedure computes the Bruhat Representation of A and return the rank of A.*
- template<class [Field](#) >  
void [getLTBruhatGen](#) (const [Field](#) &Fi, const size\_t N, const size\_t r, const size\_t \*P, const size\_t \*Q, type-name [Field::Element\\_ptr](#) R, const size\_t ldr)  
*GetLTBruhatGen This procedure Computes the Rank Revealing Matrix based on the Bruhta representation of a Matrix.*
- template<class [Field](#) >  
void [getLTBruhatGen](#) (const [Field](#) &Fi, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const [FFLAS::FFLAS\\_DIAG](#) diag, const size\_t N, const size\_t r, const size\_t \*P, const size\_t \*Q, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) T, const size\_t ldt)  
*GetLTBruhatGen This procedure computes the matrix L or U f the Bruhat Representation Suppose that A is the bruhat representation of a matrix.*
- size\_t [LTQSorder](#) (const size\_t N, const size\_t r, const size\_t \*P, const size\_t \*Q)  
*LTQSorder This procedure computes the order of quasiseparability of a matrix.*
- template<class [Field](#) >  
size\_t [CompressToBlockBiDiagonal](#) (const [Field](#) &Fi, const [FFLAS::FFLAS\\_UPLO](#) Uplo, size\_t N, size\_t s, size\_t r, const size\_t \*P, const size\_t \*Q, typename [Field::Element\\_ptr](#) A, size\_t lda, typename [Field::Element\\_ptr](#) X, size\_t ldx, size\_t \*K, size\_t \*M, size\_t \*T)  
*CompressToBlockBiDiagonal This procedure compress a compact representation of a row echelon form or column echelon form.*
- template<class [Field](#) >  
void [ExpandBlockBiDiagonalToBruhat](#) (const [Field](#) &Fi, const [FFLAS::FFLAS\\_UPLO](#) Uplo, size\_t N, size\_t s, size\_t r, typename [Field::Element\\_ptr](#) A, size\_t lda, typename [Field::Element\\_ptr](#) X, size\_t ldx, size\_t NbBlocks, size\_t \*K, size\_t \*M, size\_t \*T)

*ExpandBlockBiDiagonal* This procedure expand a compact representation of a row echelon form or column echelon form.

- void [Bruhat2EchelonPermutation](#) (size\_t N, size\_t R, const size\_t \*P, const size\_t \*Q, size\_t \*M)  
*Bruhat2EchelonPermutation (N,R,P,Q)* Compute M such that LM or MU is in echelon form where L or U are factors of the Bruhat Representation.
- size\_t \* [TInverter](#) (const size\_t \*T, size\_t r)
- template<class [Field](#) >  
void [ComputeRPermutation](#) (const [Field](#) &Fi, size\_t N, size\_t r, const size\_t \*P, const size\_t \*Q, size\_t \*R, const size\_t \*MU, const size\_t \*ML)
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [expandLCRE](#) (const [Field](#) &Fi, size\_t N, size\_t s, size\_t r, size\_t \*R, size\_t i, typename [Field::ConstElement\\_ptr](#) Xu, size\_t ldu, size\_t NbBlocksU, const size\_t \*Ku, const size\_t \*Tuinv, typename [Field::ConstElement\\_ptr](#) Xl, size\_t ldl, size\_t NbBlocksL, const size\_t \*Kl, const size\_t \*Tlinv, typename [Field::Element\\_ptr](#) CRE, size\_t ldcre)  
*Expands an anti-diagonal block of a left triangular matrix from its compact Bruhat representation.*
- template<class [Field](#) >  
void [productBruhatxTS](#) (const [Field](#) &Fi, size\_t N, size\_t s, size\_t r, size\_t t, const size\_t \*P, const size\_t \*Q, typename [Field::ConstElement\\_ptr](#) Xu, size\_t ldu, size\_t NbBlocksU, const size\_t \*Ku, const size\_t \*Tu, const size\_t \*MU, typename [Field::ConstElement\\_ptr](#) Xl, size\_t ldl, size\_t NbBlocksL, const size\_t \*Kl, const size\_t \*Tl, const size\_t \*ML, typename [Field::Element\\_ptr](#) B, size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) D, size\_t ldd)  
*Compute the product of a left-triangular quasi-separable matrix A, represented by a compact Bruhat generator, with a dense rectangular matrix B:  $C \leftarrow A \times B + \text{beta}C$ .*

## 13.182.1 Macro Definition Documentation

### 13.182.1.1 \_\_FFLASFFPACK\_ffpack\_bruhatgen\_inl

```
#define __FFLASFFPACK_ffpack_bruhatgen_inl
```

## 13.183 ffpack\_charpoly.inl File Reference

```
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include "ffpack_charpoly_mp.inl"
```

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*
- namespace [FFPACK::Protected](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_charpoly\\_INL](#)

### Functions

- template<class [PolRing](#) >  
std::list< typename [PolRing::Element](#) > & [CharPoly](#) (const [PolRing](#) &R, std::list< typename [PolRing::Element](#) > &charp, const size\_t N, typename [PolRing::Domain\\_t::Element\\_ptr](#) A, const size\_t lda, typename [PolRing::Domain\\_t::RandIter](#) &G, const [FFPACK\\_CHARPOLY\\_TAG](#) CharpTag=FpackAuto, const size\_t degree=[\\_\\_FFLASFFPACK\\_ARITHPROG\\_THRESHOLD](#))  
*Compute the characteristic polynomial of the matrix A.*
- template<class [PolRing](#) >  
[PolRing::Element](#) & [CharPoly](#) (const [PolRing](#) &R, typename [PolRing::Element](#) &charp, const size\_t N, typename [PolRing::Domain\\_t::Element\\_ptr](#) A, const size\_t lda, typename [PolRing::Domain\\_t::RandIter](#) &G, const [FFPACK\\_CHARPOLY\\_TAG](#) CharpTag=FpackAuto, const size\_t degree=[\\_\\_FFLASFFPACK\\_ARITHPROG\\_THRESHOLD](#))

*Compute the characteristic polynomial of the matrix A.*

- template<class [Field](#) , class Polynomial , class RandIter >  
std::list< Polynomial > & [LUKrylov](#) (const [Field](#) &F, std::list< Polynomial > &charp, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) U, const size\_t ldu, RandIter &G)
- template<class [Field](#) , class Polynomial >  
std::list< Polynomial > & [LUKrylov\\_KGFast](#) (const [Field](#) &F, std::list< Polynomial > &charp, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) X, const size\_t ldx)

### 13.183.1 Macro Definition Documentation

#### 13.183.1.1 \_\_FFLASFFPACK\_charpoly\_INL

```
#define __FFLASFFPACK_charpoly_INL
```

## 13.184 ffpack\_charpoly\_danilevski.inl File Reference

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [\\_\\_FFLASFFPACK\\_ffpack\\_charpoly\\_danilveski\\_INL](#)

### Functions

- template<class [Field](#) , class Polynomial >  
std::list< Polynomial > & [Danilevski](#) (const [Field](#) &F, std::list< Polynomial > &charp, const size\_t N, type-  
name [Field::Element\\_ptr](#) A, const size\_t lda)

### 13.184.1 Macro Definition Documentation

#### 13.184.1.1 \_\_FFLASFFPACK\_ffpack\_charpoly\_danilveski\_INL

```
#define __FFLASFFPACK_ffpack_charpoly_danilveski_INL
```

## 13.185 ffpack\_charpoly\_kgfast.inl File Reference

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*
- namespace [FFPACK::Protected](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_ffpack\\_charpoly\\_kgfast\\_INL](#)

### Functions

- template<class [Field](#) , class Polynomial >  
int [KGFast](#) (const [Field](#) &F, std::list< Polynomial > &charp, const size\_t N, typename [Field::Element\\_ptr](#) A,  
const size\_t lda, size\_t \*kg\_mc, size\_t \*kg\_mb, size\_t \*kg\_j)
- template<class [Field](#) >  
void [fgemv\\_kgf](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, type-  
name [Field::ConstElement\\_ptr](#) X, const size\_t incX, typename [Field::Element\\_ptr](#) Y, const size\_t incY, const  
size\_t kg\_mc, const size\_t kg\_mb, const size\_t kg\_j)

### 13.185.1 Macro Definition Documentation

#### 13.185.1.1 \_\_FflasFFPACK\_ffpack\_charpoly\_kgfast\_INL

```
#define __FflasFFPACK_ffpack_charpoly_kgfast_INL
```

### 13.186 fpack\_charpoly\_kgfastgeneralized.inl File Reference

```
#include <iostream>
#include "fflas-ffpack/utils/fflas_io.h"
```

#### Namespaces

- namespace [FFPACK](#)  
*Finite Field PACK Set of elimination based routines for dense linear algebra.*
- namespace [FFPACK::Protected](#)

#### Macros

- #define [\\_\\_FflasFFPACK\\_ffpack\\_charpoly\\_kgfastgeneralized\\_INL](#)

#### Functions

- template<class [Field](#) >  
[Field::Element\\_ptr buildMatrix](#) (const [Field](#) &F, typename [Field::ConstElement\\_ptr](#) E, typename [Field::ConstElement\\_ptr](#) C, const size\_t lda, const size\_t \*B, const size\_t \*T, const size\_t me, const size\_t mc, const size\_t lambda, const size\_t mu)
- template<class [Field](#) , class Polynomial >  
std::list< Polynomial > & [KGFast\\_generalized](#) (const [Field](#) &F, std::list< Polynomial > &charp, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda)

### 13.186.1 Macro Definition Documentation

#### 13.186.1.1 \_\_FflasFFPACK\_ffpack\_charpoly\_kgfastgeneralized\_INL

```
#define __FflasFFPACK_ffpack_charpoly_kgfastgeneralized_INL
```

### 13.187 fpack\_charpoly\_kglu.inl File Reference

#### Namespaces

- namespace [FFPACK](#)  
*Finite Field PACK Set of elimination based routines for dense linear algebra.*
- namespace [FFPACK::Protected](#)

#### Macros

- #define [\\_\\_FflasFFPACK\\_ffpack\\_charpoly\\_kglu\\_INL](#)

#### Functions

- template<class [Field](#) >  
size\_t [updateD](#) (const [Field](#) &F, size\_t \*d, size\_t k, std::vector< std::vector< typename [Field::Element](#) > > &minpt)
- template<class [Field](#) >  
size\_t [newD](#) (const [Field](#) &F, size\_t \*d, bool &KeepOn, const size\_t l, const size\_t N, typename [Field::Element\\_ptr](#) X, const size\_t \*Q, std::vector< std::vector< typename [Field::Element](#) > > &minpt)

- template<class [Field](#) , class Polynomial >  
std::list< Polynomial > & [KellerGehrig](#) (const [Field](#) &F, std::list< Polynomial > &charp, const size\_t N, type-  
name [Field::ConstElement\\_ptr](#) A, const size\_t lda)

### 13.187.1 Macro Definition Documentation

#### 13.187.1.1 \_\_FFLASFFPACK\_ffpack\_charpoly\_kglu\_INL

```
#define __FFLASFFPACK_ffpack_charpoly_kglu_INL
```

## 13.188 ffpack\_charpoly\_mp.inl File Reference

```
#include <givaro/zring.h>
#include "givaro/givinteger.h"
#include "givaro/givpoly1.h"
#include "fflas-ffpack/field/rns-integer.h"
#include "fflas-ffpack/fflas-ffpack.h"
```

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [\\_\\_FFPACK\\_charpoly\\_mp\\_INL](#)

### Functions

- [FFPACK::RNSInteger< FFPACK::rns\\_double >::Element\\_ptr CharPoly](#) (const [FFPACK::RNSInteger< FFPACK::rns\\_double > &F](#), typename [FFPACK::RNSInteger< FFPACK::rns\\_double >::Element\\_ptr](#) charp, const size\_t N, typename [FFPACK::RNSInteger< FFPACK::rns\\_double >::Element\\_ptr](#) A, const size\_t lda, Givaro::ZRing< Givaro::Integer >::Randlter &G, const FFPACK\_CHARPOLY\_TAG CharpTag, size\_t degree)
- template<> Givaro::Poly1Dom< Givaro::ZRing< Givaro::Integer > >::Element & [CharPoly](#) (const Givaro::Poly1Dom< Givaro::ZRing< Givaro::Integer > > &R, Givaro::Poly1Dom< Givaro::ZRing< Givaro::Integer > >::Element &charp, const size\_t N, Givaro::Integer \*A, const size\_t lda, Givaro::ZRing< Givaro::Integer >::Randlter &G, const FFPACK\_CHARPOLY\_TAG CharpTag, size\_t degree)

### 13.188.1 Macro Definition Documentation

#### 13.188.1.1 \_\_FFPACK\_charpoly\_mp\_INL

```
#define __FFPACK_charpoly_mp_INL
```

## 13.189 ffpack\_det\_mp.inl File Reference

```
#include <givaro/zring.h>
#include "givaro/givinteger.h"
#include "fflas-ffpack/field/rns-integer.h"
#include "fflas-ffpack/fflas-ffpack.h"
```

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

## Macros

- `#define __FFPACK_det_mp_INL`

## Functions

- `template<class PSHelper >`  
`FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr & Det (const FFPACK::RNSInteger<`  
`FFPACK::rns_double > &F, typename FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr &det,`  
`const size_t N, typename FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr A, const size_t lda,`  
`const PSHelper &psH)`
- `template<class PSHelper >`  
`Givaro::Integer & Det (const Givaro::ZRing< Givaro::Integer > &F, Givaro::Integer &det, const size_t N,`  
`Givaro::Integer *A, const size_t lda, const PSHelper &psH, size_t *P, size_t *Q)`

## 13.189.1 Macro Definition Documentation

### 13.189.1.1 \_\_FFPACK\_det\_mp\_INL

```
#define __FFPACK_det_mp_INL
```

## 13.190 ffpack\_echelonforms.inl File Reference

### Namespaces

- namespace `FFPACK`  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

## Macros

- `#define __FFLASFFPACK_ffpack_echelon_forms_INL`
- `#define __FFLASFFPACK_GAUSSJORDAN_BASECASE 256`

## Functions

- `template<class Field >`  
`void getTriangular (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag, const`  
`size_t M, const size_t N, const size_t R, typename Field::ConstElement_ptr A, const size_t lda, typename`  
`Field::Element_ptr T, const size_t ldt, const bool OnlyNonZeroVectors=false)`  
*Extracts a triangular matrix from a compact storage  $A=L\backslash U$  of rank  $R$ .*
- `template<class Field >`  
`void getTriangular (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag, const`  
`size_t M, const size_t N, const size_t R, typename Field::Element_ptr A, const size_t lda)`  
*Cleans up a compact storage  $A=L\backslash U$  to reveal a triangular matrix of rank  $R$ .*
- `void PLUQtoEchelonPermutation (const size_t N, const size_t R, const size_t *P, size_t *outPerm)`  
*Auxiliary routine: determines the permutation that changes a PLUQ decomposition into a echelon form revealing PLUQ decomposition.*
- `template<class Field >`  
`void getEchelonForm (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag,`  
`const size_t M, const size_t N, const size_t R, const size_t *P, typename Field::ConstElement_ptr A, const`  
`size_t lda, typename Field::Element_ptr T, const size_t ldt, const bool OnlyNonZeroVectors=false, const`  
`FFPACK_LU_TAG LuTag=FpackSlabRecursive)`  
*Extracts a matrix in echelon form from a compact storage  $A=L\backslash U$  of rank  $R$  obtained by RowEchelonForm or Column↔EchelonForm.*
- `template<class Field >`  
`void getEchelonForm (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag,`  
`const size_t M, const size_t N, const size_t R, const size_t *P, typename Field::Element_ptr A, const size_t`  
`lda, const FFPACK_LU_TAG LuTag=FpackSlabRecursive)`



*Cleans up a compact storage  $A=L\backslash U$  obtained by RowEchelonForm or ColumnEchelonForm to reveal an echelon form of rank  $R$ .*

- template<class [Field](#) >  
void [getEchelonTransform](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const [FFLAS::FFLAS\\_DIAG](#) diag, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const size\_t \*Q, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) T, const size\_t ldt, const [FFPACK\\_LU\\_TAG](#) LuTag=[FfpackSlabRecursive](#))

*Extracts a transformation matrix to echelon form from a compact storage  $A=L\backslash U$  of rank  $R$  obtained by RowEchelonForm or ColumnEchelonForm.*

- template<class [Field](#) >  
void [getReducedEchelonForm](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) T, const size\_t ldt, const bool OnlyNonZeroVectors=false, const [FFPACK\\_LU\\_TAG](#) LuTag=[FfpackSlabRecursive](#))

*Extracts a matrix in echelon form from a compact storage  $A=L\backslash U$  of rank  $R$  obtained by ReducedRowEchelonForm or ReducedColumnEchelonForm with transform = true.*

- template<class [Field](#) >  
void [getReducedEchelonForm](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, typename [Field::Element\\_ptr](#) A, const size\_t lda, const [FFPACK\\_LU\\_TAG](#) LuTag=[FfpackSlabRecursive](#))

*Cleans up a compact storage  $A=L\backslash U$  of rank  $R$  obtained by ReducedRowEchelonForm or ReducedColumnEchelonForm with transform = true.*

- template<class [Field](#) >  
void [getReducedEchelonTransform](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const size\_t \*Q, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) T, const size\_t ldt, const [FFPACK\\_LU\\_TAG](#) LuTag=[FfpackSlabRecursive](#))

*Extracts a transformation matrix to echelon form from a compact storage  $A=L\backslash U$  of rank  $R$  obtained by RowEchelonForm or ColumnEchelonForm.*

## 13.190.1 Macro Definition Documentation

### 13.190.1.1 \_\_FFLASFFPACK\_ffpack\_echelon\_forms\_INL

```
#define __FFLASFFPACK_ffpack_echelon_forms_INL
```

### 13.190.1.2 \_\_FFLASFFPACK\_GAUSSJORDAN\_BASECASE

```
#define __FFLASFFPACK_GAUSSJORDAN_BASECASE 256
```

## 13.191 ffpack\_fgesv.inl File Reference

### Namespaces

- namespace [FFPACK](#)  
*Finite **Field** **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [\\_\\_FFLASFFPACK\\_ffpack\\_fgesv\\_INL](#)

### Functions

- template<class [Field](#) >  
size\_t [fgesv](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) B, const size\_t ldb, int \*info)  
*Square system solver.*

- template<class [Field](#) >  
size\_t [fgesv](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t NRHS, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) X, const size\_t ldx, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, int \*info)

*Rectangular system solver.*

## 13.191.1 Macro Definition Documentation

### 13.191.1.1 \_\_FFLASFFPACK\_ffpack\_fgesv\_INL

```
#define __FFLASFFPACK_ffpack_fgesv_INL
```

## 13.192 ffpack\_fgetrs.inl File Reference

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [\\_\\_FFLASFFPACK\\_ffpack\\_fgetrs\\_INL](#)

### Functions

- template<class [Field](#) >  
void [fgetrs](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t R, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t \*P, const size\_t \*Q, typename [Field::Element\\_ptr](#) B, const size\_t ldb, int \*info)  
*Solve the system  $AX = B$  or  $XA = B$ .*
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [fgetrs](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t NRHS, const size\_t R, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t \*P, const size\_t \*Q, typename [Field::Element\\_ptr](#) X, const size\_t ldx, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, int \*info)  
*Solve the system  $A X = B$  or  $X A = B$ .*

## 13.192.1 Macro Definition Documentation

### 13.192.1.1 \_\_FFLASFFPACK\_ffpack\_fgetrs\_INL

```
#define __FFLASFFPACK_ffpack_fgetrs_INL
```

## 13.193 ffpack\_frobenius.inl File Reference

```
#include <givaro/givranditer.h>
```

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*
- namespace [FFPACK::Protected](#)

## Functions

- template<class [Field](#) >  
void [CompressRows](#) ([Field](#) &F, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) tmp, const size\_t ldtmp, const size\_t \*d, const size\_t nb\_blocs)
- template<class [Field](#) >  
void [CompressRowsQK](#) ([Field](#) &F, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) tmp, const size\_t ldtmp, const size\_t \*d, const size\_t deg, const size\_t nb\_blocs)
- template<class [Field](#) >  
void [DeCompressRows](#) ([Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) tmp, const size\_t ldtmp, const size\_t \*d, const size\_t nb\_blocs)
- template<class [Field](#) >  
void [DeCompressRowsQK](#) ([Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) tmp, const size\_t ldtmp, const size\_t \*d, const size\_t deg, const size\_t nb\_blocs)
- template<class [Field](#) >  
void [CompressRowsQA](#) ([Field](#) &F, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) tmp, const size\_t ldtmp, const size\_t \*d, const size\_t nb\_blocs)
- template<class [Field](#) >  
void [DeCompressRowsQA](#) ([Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) tmp, const size\_t ldtmp, const size\_t \*d, const size\_t nb\_blocs)
- template<class [PolRing](#) >  
void [RandomKrylovPrecond](#) (const [PolRing](#) &PR, std::list< typename [PolRing::Element](#) > &completedFactors, const size\_t N, typename [PolRing::Domain\\_t::Element\\_ptr](#) A, const size\_t lda, size\_t &Nb, typename [PolRing::Domain\\_t::Element\\_ptr](#) &B, size\_t &ldb, typename [PolRing::Domain\\_t::RandIter](#) &g, const size\_t degree=\_\_FFLASFFPACK\_ARITHPROG\_THRESHOLD)
- template<class [PolRing](#) >  
std::list< typename [PolRing::Element](#) > & [ArithProg](#) (const [PolRing](#) &PR, std::list< typename [PolRing::Element](#) > &frobeniusForm, const size\_t N, typename [PolRing::Domain\\_t::Element\\_ptr](#) A, const size\_t lda, const size\_t degree)

## 13.194 ffpack\_fsytrf.inl File Reference

```
#include "fflas-ffpack/utils/fflas_io.h"
```

## Namespaces

- namespace [FFPACK](#)  
*Finite **Field** **PACK** Set of elimination based routines for dense linear algebra.*

## Macros

- #define [\\_\\_FFLASFFPACK\\_ffpack\\_fsytrf\\_INL](#)

## Functions

- template<class [Field](#) >  
bool [fsytrf\\_BC\\_Crout](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) UpLo, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) Din, const size\_t incDinv)
- template<class [Field](#) >  
size\_t [fsytrf\\_BC\\_RL](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) UpLo, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) Din, const size\_t incDinv)
- template<class [Field](#) >  
size\_t [fsytrf\\_UP\\_RPM\\_BC\\_RL](#) (const [Field](#) &F, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) Din, const size\_t incDinv, size\_t \*P)

- `template<class Field >`  
`size_t fsytrf_LOW_RPM_BC_Crout (const Field &F, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr Dinv, const size_t incDinv, size_t *P)`
- `template<class Field >`  
`size_t fsytrf_UP_RPM_BC_Crout (const Field &F, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr Dinv, const size_t incDinv, size_t *P)`
- `template<class Field >`  
`size_t fsytrf_UP_RPM (const Field &F, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr Dinv, const size_t incDinv, size_t *P, size_t BCThreshold)`
- `template<class Field >`  
`bool fsytrf_nonunit (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr Dinv, const size_t incDinv, FFLAS::ParSeqHelper::Sequential seq, size_t threshold)`
- `template<class Field, class Cut, class Param >`  
`bool fsytrf_nonunit (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr Dinv, const size_t incDinv, FFLAS::ParSeqHelper::Parallel< Cut, Param > par, size_t threshold)`
- `template<class Field >`  
`bool fsytrf (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, const size_t threshold=__FFLASFFPACK_FSYTRF_THRESHOLD)`  
*Triangular factorization of symmetric matrices.*
- `template<class Field >`  
`bool fsytrf (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, const FFLAS::ParSeqHelper::Sequential seq, const size_t threshold=__FFLASFFPACK_FSYTRF_THRESHOLD)`
- `template<class Field, class Cut, class Param >`  
`bool fsytrf (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, const FFLAS::ParSeqHelper::Parallel< Cut, Param > par, const size_t threshold=__FFLASFFPACK_FSYTRF_THRESHOLD)`
- `template<class Field >`  
`size_t fsytrf_RPM (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t threshold)`
- `template<class Field >`  
`void getTridiagonal (const Field &F, const size_t N, const size_t R, typename Field::ConstElement_ptr A, const size_t lda, size_t *P, typename Field::Element_ptr T, const size_t ldt)`

### 13.194.1 Macro Definition Documentation

#### 13.194.1.1 \_\_FFLASFFPACK\_ffpack\_fsytrf\_INL

```
#define __FFLASFFPACK_ffpack_fsytrf_INL
```

## 13.195 ffpack\_ftrssyr2k.inl File Reference

```
#include "fflas-ffpack/utils/fflas_io.h"
```

### Namespaces

- namespace **FFPACK**  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define **\_\_FFLASFFPACK\_ffpack\_ftrssyr2k\_INL**

## Functions

- `template<class Field >`  
`void ftrssyr2k (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diagA, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, const size_t threshold=__FFLASFFPACK_FTRSSYR2K_THRESHOLD)`  
*Solve a triangular system in a symmetric sum: find B upper/lower triangular such that  $A^T B + B^T A = C$  where C is symmetric.*

## 13.195.1 Macro Definition Documentation

### 13.195.1.1 \_\_FFLASFFPACK\_ffpack\_ftrssyr2k\_INL

```
#define __FFLASFFPACK_ffpack_ftrssyr2k_INL
```

## 13.196 ffpack\_ftrstr.inl File Reference

```
#include "fflas-ffpack/utils/fflas_io.h"
```

## Namespaces

- namespace `FFPACK`  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

## Macros

- `#define __FFLASFFPACK_ffpack_ftrstr_INL`

## Functions

- `template<class Field >`  
`void ftrstr (const Field &F, const FFLAS::FFLAS_SIDE side, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diagA, const FFLAS::FFLAS_DIAG diagB, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, const size_t threshold=__FFLASFFPACK_FTRSTR_THRESHOLD)`  
*Solve a triangular system with a triangular right hand side of the same shape.*

## 13.196.1 Macro Definition Documentation

### 13.196.1.1 \_\_FFLASFFPACK\_ffpack\_ftrstr\_INL

```
#define __FFLASFFPACK_ffpack_ftrstr_INL
```

## 13.197 ffpack\_ftrtr.inl File Reference

## Namespaces

- namespace `FFPACK`  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

## Macros

- `#define ENABLE_ALL_CHECKINGS 1`
- `#define __FFLASFFPACK_ffpack_ftrtr_INL`

## Functions

- template<class [Field](#) >  
void [ftrtri](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t threshold=[\\_\\_FFLASFFPACK\\_FTRTRI\\_THRESHOLD](#))  
*Compute the inverse of a triangular matrix.*
- template<class [Field](#) >  
void [ftrrm](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) side, const [FFLAS::FFLAS\\_DIAG](#) diag, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*Compute the product of two triangular matrices of opposite shape.*
- template<class [Field](#) >  
void [trinv\\_left](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) L, const size\_t ldl, typename [Field::Element\\_ptr](#) X, const size\_t ldx)

## 13.197.1 Macro Definition Documentation

### 13.197.1.1 [ENABLE\\_ALL\\_CHECKINGS](#)

```
#define ENABLE\_ALL\_CHECKINGS 1
```

### 13.197.1.2 [\\_\\_FFLASFFPACK\\_ffpack\\_ftrtr\\_INL](#)

```
#define \_\_FFLASFFPACK\_ffpack\_ftrtr\_INL
```

## 13.198 [ffpack\\_invert.inl](#) File Reference

### Namespaces

- namespace [FFPACK](#)  
*Finite **Field** **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [\\_\\_FFLASFFPACK\\_ffpack\\_invert\\_INL](#)

### Functions

- template<class [Field](#) >  
[Field::Element\\_ptr](#) [Invert](#) (const [Field](#) &F, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t lda, int &>nullity)  
*Invert the given matrix in place or computes its nullity if it is singular.*
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [Invert](#) (const [Field](#) &F, const size\_t M, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) X, const size\_t ldx, int &>nullity)  
*Invert the given matrix or computes its nullity if it is singular.*
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [Invert2](#) (const [Field](#) &F, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) X, const size\_t ldx, int &>nullity)  
*Invert the given matrix or computes its nullity if it is singular.*

## 13.198.1 Macro Definition Documentation

### 13.198.1.1 [\\_\\_FFLASFFPACK\\_ffpack\\_invert\\_INL](#)

```
#define \_\_FFLASFFPACK\_ffpack\_invert\_INL
```

## 13.199 ffpack\_krylovelim.inl File Reference

### Macros

- `#define __FFLASFFPACK_ffpack_krylovelim_INL`

### 13.199.1 Macro Definition Documentation

#### 13.199.1.1 \_\_FFLASFFPACK\_ffpack\_krylovelim\_INL

```
#define __FFLASFFPACK_ffpack_krylovelim_INL
```

## 13.200 ffpack\_ludivine.inl File Reference

```
#include "fflas-ffpack/fflas/fflas_bounds.inl"
```

### Data Structures

- class `callLUdivine_small< Element >`
- class `callLUdivine_small< double >`
- class `callLUdivine_small< float >`

### Namespaces

- namespace `FFPACK`  
*Finite Field PACK Set of elimination based routines for dense linear algebra.*
- namespace `FFPACK::Protected`

### Macros

- `#define __FFLASFFPACK_ffpack_ludivine_INL`

### Functions

- `template<class Field >`  
`size_t LUdivine_gauss (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, const FFPACK::FFPACK_LU_TAG LuTag)`
- `template<class Field >`  
`size_t LUdivine_small (const Field &F, const FFLAS::FFLAS_DIAG Diag, const FFLAS::FFLAS_TRANSPOSE trans, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, const FFPACK::FFPACK_LU_TAG LuTag)`
- `template<class Field >`  
`size_t LUdivine (const Field &F, const FFLAS::FFLAS_DIAG Diag, const FFLAS::FFLAS_TRANSPOSE trans, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, const FFPACK::FFPACK_LU_TAG LuTag, const size_t cutoff)`
- `template<class Field >`  
`size_t LUdivine_construct (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr X, const size_t idx, typename Field::Element_ptr u, const size_t incu, size_t *P, bool computeX, const FFPACK::FFPACK_MINPOLY_TAG MinTag, const size_t kg_mc, const size_t kg_mb, const size_t kg_j)`

### 13.200.1 Macro Definition Documentation

#### 13.200.1.1 \_\_FFLASFFPACK\_ffpack\_ludivine\_INL

```
#define __FFLASFFPACK_ffpack_ludivine_INL
```

## 13.201 ffpack\_ludivine\_mp.inl File Reference

```
#include <givaro/modular-integer.h>
#include <givaro/givinteger.h>
#include "fflas-ffpack/field/rns-integer-mod.h"
#include "fflas-ffpack/field/rns-integer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/ffpack/ffpack_ludivine.inl"
```

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [\\_\\_FFPACK\\_ludivine\\_mp\\_INL](#)

### Functions

- template<> size\_t [LUdivine](#) (const Givaro::Modular< Givaro::Integer > &F, const [FFLAS::FFLAS\\_DIAG](#) Diag, const [FFLAS::FFLAS\\_TRANSPOSE](#) trans, const size\_t M, const size\_t N, typename Givaro::Integer \*A, const size\_t lda, size\_t \*P, size\_t \*Q, const FFPACK::FFPACK\_LU\_TAG LuTag, const size\_t cutoff)

## 13.201.1 Macro Definition Documentation

### 13.201.1.1 \_\_FFPACK\_ludivine\_mp\_INL

```
#define __FFPACK_ludivine_mp_INL
```

## 13.202 ffpack\_minpoly.inl File Reference

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*
- namespace [FFPACK::Protected](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_ffpack\\_minpoly\\_INL](#)

### Functions

- template<class [Field](#) , class Polynomial >  
Polynomial & [MinPoly](#) (const [Field](#) &F, Polynomial &minP, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda)  
*Compute the minimal polynomial of the matrix A.*
- template<class [Field](#) , class Polynomial , class RandIter >  
Polynomial & [MinPoly](#) (const [Field](#) &F, Polynomial &minP, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, RandIter &G)  
*Compute the minimal polynomial of the matrix A.*
- template<class [Field](#) , class Polynomial >  
Polynomial & [MatVecMinPoly](#) (const [Field](#) &F, Polynomial &minP, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) v, const size\_t incv)  
*Compute the minimal polynomial of the matrix A and a vector v, namely the first linear dependency relation in the Krylov basis  $(v, Av, \dots, A^N v)$ .*



- template<class [Field](#) , class Polynomial >  
Polynomial & [MatVecMinPoly](#) (const [Field](#) &F, Polynomial &minP, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) v, const size\_t incv, typename [Field::Element\\_ptr](#) K, const size\_t ldk, size\_t \*P)
- template<class [Field](#) , class Polynomial >  
Polynomial & [Hybrid\\_KGF\\_LUK\\_MinPoly](#) (const [Field](#) &F, Polynomial &minP, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) X, const size\_t ldx, size\_t \*P, const FFPACK\_MINPOLY\_TAG MinTag=FFPACK::FfpackDense, const size\_t kg\_mc=0, const size\_t kg\_←mb=0, const size\_t kg\_j=0)

## 13.202.1 Macro Definition Documentation

### 13.202.1.1 \_\_FFLASFFPACK\_ffpack\_minpoly\_INL

```
#define __FFLASFFPACK_ffpack_minpoly_INL
```

## 13.203 ffpack\_permutation.inl File Reference

```
#include <givaro/zring.h>
#include "fflas-ffpack/fflas/fflas_fassign.h"
```

### Namespaces

- namespace [FFPACK](#)  
*Finite **Field** **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [\\_\\_FFLASFFPACK\\_ffpack\\_permutation\\_INL](#)
- #define [FFLASFFPACK\\_PERM\\_BKSIZE](#) 32

### Functions

- template<class [Field](#) >  
void [MonotonicApplyP](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const [FFLAS::FFLAS\\_TRANSPOSE](#) Trans, const size\_t M, const size\_t ibeg, const size\_t iend, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t \*P, const size\_t R)
- Apply a R-monotonically increasing permutation P, to the matrix A.*
- template<class [Field](#) >  
void [MonotonicCompress](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t incA, const size\_t \*MathP, const size\_t R, const size\_t maxpiv, const size\_t rowstomove, const std::vector< bool > &ispiv)
- template<class [Field](#) >  
void [MonotonicCompressMorePivots](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t incA, const size\_t \*MathP, const size\_t R, const size\_t rowstomove, const size\_t lenP)
- template<class [Field](#) >  
void [MonotonicCompressCycles](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t incA, const size\_t \*MathP, const size\_t lenP)
- template<class [Field](#) >  
void [MonotonicExpand](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t incA, const size\_t \*MathP, const size\_t R, const size\_t maxpiv, const size\_t rowstomove, const std::vector< bool > &ispiv)
- template<class [Field](#) >  
void [applyP\\_block](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const [FFLAS::FFLAS\\_TRANSPOSE](#) Trans, const size\_t M, const size\_t ibeg, const size\_t iend, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t \*P)

- `template<class Field >`  
`void applyP (const Field &F, const FFLAS::FFLAS_SIDE Side, const FFLAS::FFLAS_TRANSPOSE Trans,`  
`const size_t m, const size_t ibeg, const size_t iend, typename Field::Element_ptr A, const size_t lda, const`  
`size_t *P, const FFLAS::ParSeqHelper::Sequential seq)`
- `template<class Field >`  
`void doApplyS (const Field &F, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr`  
`tmp, const size_t width, const size_t M2, const size_t R1, const size_t R2, const size_t R3, const size_t R4)`
- `template<class Field >`  
`void MatrixApplyS (const Field &F, typename Field::Element_ptr A, const size_t lda, const size_t width, const`  
`size_t M2, const size_t R1, const size_t R2, const size_t R3, const size_t R4)`
- `template<class Field >`  
`void MatrixApplyS (const Field &F, typename Field::Element_ptr A, const size_t lda, const size_t ↵`  
`t width, const size_t M2, const size_t R1, const size_t R2, const size_t R3, const size_t R4, const`  
`FFLAS::ParSeqHelper::Sequential seq)`
- `template<class Field, class Cut, class Param >`  
`void MatrixApplyS (const Field &F, typename Field::Element_ptr A, const size_t lda, const size_t ↵`  
`t width, const size_t M2, const size_t R1, const size_t R2, const size_t R3, const size_t R4, const`  
`FFLAS::ParSeqHelper::Parallel< Cut, Param > par)`
- `template<class T >`  
`void PermApplyS (T *A, const size_t lda, const size_t width, const size_t M2, const size_t R1, const size_t`  
`R2, const size_t R3, const size_t R4)`
- `template<class Field >`  
`void doApplyT (const Field &F, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr`  
`tmp, const size_t width, const size_t N2, const size_t R1, const size_t R2, const size_t R3, const size_t R4)`
- `template<class Field >`  
`void MatrixApplyT (const Field &F, typename Field::Element_ptr A, const size_t lda, const size_t width, const`  
`size_t N2, const size_t R1, const size_t R2, const size_t R3, const size_t R4)`
- `template<class Field >`  
`void MatrixApplyT (const Field &F, typename Field::Element_ptr A, const size_t lda, const size_t ↵`  
`t width, const size_t N2, const size_t R1, const size_t R2, const size_t R3, const size_t R4, const`  
`FFLAS::ParSeqHelper::Sequential seq)`
- `template<class Field, class Cut, class Param >`  
`void MatrixApplyT (const Field &F, typename Field::Element_ptr A, const size_t lda, const size_t ↵`  
`t width, const size_t N2, const size_t R1, const size_t R2, const size_t R3, const size_t R4, const`  
`FFLAS::ParSeqHelper::Parallel< Cut, Param > par)`
- `template<class T >`  
`void PermApplyT (T *A, const size_t lda, const size_t width, const size_t N2, const size_t R1, const size_t`  
`R2, const size_t R3, const size_t R4)`
- `void LAPACKPerm2MathPerm (size_t *MathP, const size_t *LapackP, const size_t N)`  
*Conversion of a permutation from LAPACK format to Math format.*
- `void MathPerm2LAPACKPerm (size_t *LapackP, const size_t *MathP, const size_t N)`  
*Conversion of a permutation from Maths format to LAPACK format.*
- `void composePermutationsLLL (size_t *P1, const size_t *P2, const size_t R, const size_t N)`  
*Computes  $P1 \times \text{Diag}(I_R, P2)$  where  $P1$  is a LAPACK and  $P2$  a LAPACK permutation and store the result in  $P1$  as a LAPACK permutation.*
- `void composePermutationsLLM (size_t *MathP, const size_t *P1, const size_t *P2, const size_t R, const`  
`size_t N)`  
*Computes  $P1 \times \text{Diag}(I_R, P2)$  where  $P1$  is a LAPACK and  $P2$  a LAPACK permutation and store the result in  $\text{MathP}$  as a MathPermutation format.*
- `void composePermutationsMLM (size_t *MathP1, const size_t *P2, const size_t R, const size_t N)`  
*Computes  $\text{MathP1} \times \text{Diag}(I_R, P2)$  where  $\text{MathP1}$  is a MathPermutation and  $P2$  a LAPACK permutation and store the result in  $\text{MathP1}$  as a MathPermutation format.*
- `void cyclic_shift_mathPerm (size_t *P, const size_t s)`
- `template<class Field >`  
`void cyclic_shift_row_col (const Field &F, typename Field::Element_ptr A, size_t m, size_t n, size_t lda)`

- template<class [Field](#) >  
void [cyclic\\_shift\\_row](#) (const [Field](#) &F, typename [Field::Element\\_ptr](#) A, size\_t m, size\_t n, size\_t lda)
- template<typename T >  
void [cyclic\\_shift\\_row](#) (const [RNSIntegerMod](#)< T > &F, typename T::Element\_ptr A, size\_t m, size\_t n, size\_t lda)
- template<class [Field](#) >  
void [cyclic\\_shift\\_col](#) (const [Field](#) &F, typename [Field::Element\\_ptr](#) A, size\_t m, size\_t n, size\_t lda)
- template<typename T >  
void [cyclic\\_shift\\_col](#) (const [RNSIntegerMod](#)< T > &F, typename T::Element\_ptr A, size\_t m, size\_t n, size\_t lda)
- template<class [Field](#) >  
void [applyP](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const [FFLAS::FFLAS\\_TRANSPOSE](#) Trans, const size\_t M, const size\_t ibeg, const size\_t iend, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t \*P)  
*Computes  $P1 \times \text{Diag}(L_R, P2)$  where  $P1$  is a LAPACK and  $P2$  a LAPACK permutation and store the result in  $P1$  as a LAPACK permutation.*
- template<class [Field](#) , class Cut , class Param >  
void [applyP](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const [FFLAS::FFLAS\\_TRANSPOSE](#) Trans, const size\_t m, const size\_t ibeg, const size\_t iend, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t \*P, const [FFLAS::ParSeqHelper::Parallel](#)< Cut, Param > par)

### 13.203.1 Macro Definition Documentation

#### 13.203.1.1 \_\_FFLASFFPACK\_ffpack\_permutation\_INL

```
#define __FFLASFFPACK_ffpack_permutation_INL
```

#### 13.203.1.2 FFLASFFPACK\_PERM\_BKSIZE

```
#define FFLASFFPACK_PERM_BKSIZE 32
```

## 13.204 ffpack\_pluq.inl File Reference

### Namespaces

- namespace [FFPACK](#)  
*Finite **Field** **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [\\_\\_FFLASFFPACK\\_ffpack\\_pluq\\_INL](#)
- #define [CROUT](#)

### Functions

- template<class [Field](#) >  
size\_t [PLUQ\\_basecaseV3](#) (const [Field](#) &Fi, const [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, typename [Field::Element](#) \*A, const size\_t lda, size\_t \*P, size\_t \*Q)
- template<class [Field](#) >  
size\_t [PLUQ\\_basecaseV2](#) (const [Field](#) &Fi, const [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, typename [Field::Element](#) \*A, const size\_t lda, size\_t \*P, size\_t \*Q)
- template<class [Field](#) >  
size\_t [PLUQ\\_basecaseCROUT](#) (const [Field](#) &Fi, const [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*P, size\_t \*Q)
- template<class [Field](#) >  
size\_t [\\_PLUQ](#) (const [Field](#) &Fi, const [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*P, size\_t \*Q, size\_t BCThreshold)

- `template<class Field >`  
`size_t PLUQ (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, type-`  
`name Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, const FFLAS::ParSeqHelper::Sequential`  
`&PSHelper, size_t BCThreshold=__FFLASFFPACK_PLUQ_THRESHOLD)`
- `template<class Field >`  
`size_t PLUQ (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename`  
`Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q)`

*Compute a PLUQ factorization of the given matrix.*

### 13.204.1 Macro Definition Documentation

#### 13.204.1.1 \_\_FFLASFFPACK\_ffpack\_pluq\_INL

```
#define __FFLASFFPACK_ffpack_pluq_INL
```

#### 13.204.1.2 CROUT

```
#define CROUT
```

## 13.205 ffpack\_pluq\_mp.inl File Reference

```
#include "fflas-ffpack/field/rns-integer-mod.h"
#include "fflas-ffpack/field/rns-integer.h"
#include "fflas-ffpack/fflas-ffpack.h"
#include "givaro/givinteger.h"
#include "givaro/modular-integer.h"
```

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- `#define __FFPACK_pluq_mp_INL`

### Functions

- `template<class Cut , class Param >`  
`size_t PLUQ (const Givaro::Modular< Givaro::Integer > &F, const FFLAS::FFLAS_DIAG Diag, const size_t`  
`M, const size_t N, typename Givaro::Integer *A, const size_t lda, size_t *P, size_t *Q, size_t BCThreshold,`  
`FFLAS::ParSeqHelper::Parallel< Cut, Param > &PSHelper)`

### 13.205.1 Macro Definition Documentation

#### 13.205.1.1 \_\_FFPACK\_pluq\_mp\_INL

```
#define __FFPACK_pluq_mp_INL
```

## 13.206 ffpack\_ppluq.inl File Reference

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

## Macros

- `#define __FFLASFFPACK_ffpack_ppluq_INL`
- `#define __FFLAS__TRSM_READONLY`
- `#define PBASECASE_K 256`

## Functions

- `template<class Field >`  
`void threads_fgemm (const size_t m, const size_t n, const size_t r, int nbthreads, size_t *W1, size_t *W2, size_t *W3, size_t gamma)`
- `template<class Field >`  
`void threads_ftrsm (const size_t m, const size_t n, int nbthreads, size_t *t1, size_t *t2)`
- `template<class Field >`  
`size_t PLUQ (const Field &Fi, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, const FFLAS::ParSeqHelper::Parallel< FFLAS::CuttingStrategy::Recursive, FFLAS::StrategyParameter::Threads > &PSHelper)`
- `template<class Field >`  
`size_t pPLUQ (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q)`

### 13.206.1 Macro Definition Documentation

#### 13.206.1.1 \_\_FFLASFFPACK\_ffpack\_ppluq\_INL

```
#define __FFLASFFPACK_ffpack_ppluq_INL
```

#### 13.206.1.2 \_\_FFLAS\_\_TRSM\_READONLY

```
#define __FFLAS__TRSM_READONLY
```

#### 13.206.1.3 PBASECASE\_K

```
#define PBASECASE_K 256
```

## 13.207 ffpack\_rankprofiles.inl File Reference

### Namespaces

- namespace `FFPACK`  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- `#define __FFLASFFPACK_ffpack_rank_profiles_INL`

### Functions

- `template<class Field >`  
`size_t RowRankProfile (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *&rkprofile, const FFPACK_LU_TAG LuTag=FfpackSlabRecursive)`  
*Computes the row rank profile of A.*
- `template<class Field >`  
`size_t pRowRankProfile (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *&rkprofile, size_t numthreads=0, const FFPACK_LU_TAG LuTag=FfpackTileRecursive)`
- `template<class Field, class PSHelper >`  
`size_t RowRankProfile (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *&rkprofile, const FFPACK_LU_TAG LuTag, PSHelper &psH)`

- `template<class Field >`  
`size_t ColumnRankProfile (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A,`  
`const size_t lda, size_t *rkprofile, const FFPACK_LU_TAG LuTag=FfpackSlabRecursive)`  
*Computes the column rank profile of A.*
- `template<class Field >`  
`size_t pColumnRankProfile (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A,`  
`const size_t lda, size_t *rkprofile, size_t numthreads=0, const FFPACK_LU_TAG LuTag=FfpackTileRecursive)`
- `template<class Field , class PSHelper >`  
`size_t ColumnRankProfile (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A,`  
`const size_t lda, size_t *rkprofile, const FFPACK_LU_TAG LuTag, PSHelper &psH)`
- `void RankProfileFromLU (const size_t *P, const size_t N, const size_t R, size_t *rkprofile, const FFPACK_LU_TAG LuTag)`  
*Recovers the column/row rank profile from the permutation of an LU decomposition.*
- `size_t LeadingSubmatrixRankProfiles (const size_t M, const size_t N, const size_t R, const size_t LSm, const`  
`size_t LSn, const size_t *P, const size_t *Q, size_t *RRP, size_t *CRP)`  
*Recovers the row and column rank profiles of any leading submatrix from the PLUQ decomposition.*
- `template<class Field >`  
`size_t RowRankProfileSubmatrixIndices (const Field &F, const size_t M, const size_t N, typename`  
`Field::Element_ptr A, const size_t lda, size_t *rowindices, size_t *colindices, size_t &R)`  
*RowRankProfileSubmatrixIndices.*
- `template<class Field >`  
`size_t ColRankProfileSubmatrixIndices (const Field &F, const size_t M, const size_t N, typename`  
`Field::Element_ptr A, const size_t lda, size_t *rowindices, size_t *colindices, size_t &R)`  
*Computes the indices of the submatrix  $r \times r$  X of A whose columns correspond to the column rank profile of A.*
- `template<class Field >`  
`size_t RowRankProfileSubmatrix (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr`  
`A, const size_t lda, typename Field::Element_ptr &X, size_t &R)`  
*Computes the  $r \times r$  submatrix X of A, by picking the row rank profile rows of A.*
- `template<class Field >`  
`size_t ColRankProfileSubmatrix (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr`  
`A, const size_t lda, typename Field::Element_ptr &X, size_t &R)`  
*Compute the  $r \times r$  submatrix X of A, by picking the row rank profile rows of A.*
- `template<class Field >`  
`Field::Element_ptr LQUPtoInverseOfFullRankMinor (const Field &F, const size_t rank, typename`  
`Field::Element_ptr A_factors, const size_t lda, const size_t *QtPointer, typename Field::Element_ptr X,`  
`const size_t ldx)`  
*LQUPtoInverseOfFullRankMinor.*

## 13.207.1 Macro Definition Documentation

### 13.207.1.1 \_\_FFLASFFPACK\_ffpack\_rank\_profiles\_INL

```
#define __FFLASFFPACK_ffpack_rank_profiles_INL
```

## 13.208 field-traits.h File Reference

Field Traits.

```
#include <type_traits>
#include "fflas-ffpack/field/rns-double-elt.h"
#include "recint/rmint.h"
#include "givaro/modular-general.h"
#include "givaro/zring.h"
```

## Data Structures

- struct [GenericTag](#)  
*generic ring.*
- struct [ModularTag](#)  
*This is a modular field like e.g. `Modular<T>` or `ModularBalanced<T>`*
- struct [UnparametricTag](#)  
*If the field uses a representation with infix operators.*
- struct [DefaultTag](#)  
*No specific mode of action: use standard field operations.*
- struct [DefaultBoundedTag](#)  
*Use standard field operations, but keeps track of bounds on input and output.*
- struct [ConvertTo< T >](#)  
*Force conversion to appropriate element type of `ElementCategory T`.*
- struct [DelayedTag](#)  
*Performs field operations with delayed mod reductions. Ensures result is reduced.*
- struct [LazyTag](#)  
*Performs field operations with delayed mod only when necessary. Result may not be reduced.*
- struct [GenericTag](#)  
*default is generic*
- struct [MachineFloatTag](#)  
*float or double*
- struct [MachineIntTag](#)  
*short, int, long, long long, and unsigned variants*
- struct [FixedPrecIntTag](#)  
*Fixed precision integers above machine precision: `Givaro::reclnt`.*
- struct [ArbitraryPrecIntTag](#)  
*Arbitrary precision integers: `GMP`.*
- struct [RNSElementTag](#)  
*Representation in a Residue Number System.*
- struct [ElementTraits< Element >](#)  
*[ElementTraits](#).*
- struct [ElementTraits< float >](#)
- struct [ElementTraits< double >](#)
- struct [ElementTraits< int8\\_t >](#)
- struct [ElementTraits< int16\\_t >](#)
- struct [ElementTraits< int32\\_t >](#)
- struct [ElementTraits< int64\\_t >](#)
- struct [ElementTraits< uint8\\_t >](#)
- struct [ElementTraits< uint16\\_t >](#)
- struct [ElementTraits< uint32\\_t >](#)
- struct [ElementTraits< uint64\\_t >](#)
- struct [ElementTraits< Givaro::Integer >](#)
- struct [ElementTraits< Reclnt::rint< K > >](#)
- struct [ElementTraits< Reclnt::ruint< K > >](#)
- struct [ElementTraits< Reclnt::rmint< K, MG > >](#)
- struct [ElementTraits< FFPACK::rns\\_double\\_elt >](#)
- struct [ModeTraits< Field >](#)  
*[ModeTraits](#).*
- struct [ModeTraits< Givaro::Modular< Element, Compute > >](#)
- struct [ModeTraits< Givaro::Modular< int64\\_t, uint64\\_t > >](#)
- struct [ModeTraits< Givaro::Modular< int8\\_t, Compute > >](#)
- struct [ModeTraits< Givaro::Modular< int16\\_t, Compute > >](#)

- struct [ModeTraits](#)< Givaro::Modular< int32\_t, Compute > >
- struct [ModeTraits](#)< Givaro::Modular< uint8\_t, Compute > >
- struct [ModeTraits](#)< Givaro::Modular< uint16\_t, Compute > >
- struct [ModeTraits](#)< Givaro::Modular< uint32\_t, Compute > >
- struct [ModeTraits](#)< Givaro::Modular< Givaro::Integer, Compute > >
- struct [ModeTraits](#)< Givaro::Modular< ReclInt::ruint< K >, Compute > >
- struct [ModeTraits](#)< Givaro::ModularBalanced< Element > >
- struct [ModeTraits](#)< Givaro::ModularBalanced< int8\_t > >
- struct [ModeTraits](#)< Givaro::ModularBalanced< int16\_t > >
- struct [ModeTraits](#)< Givaro::ModularBalanced< int32\_t > >
- struct [ModeTraits](#)< Givaro::ModularBalanced< Givaro::Integer > >
- struct [ModeTraits](#)< Givaro::ZRing< Givaro::Integer > >
- struct [ModeTraits](#)< Givaro::ZRing< float > >
- struct [ModeTraits](#)< Givaro::ZRing< double > >
- struct [ModeTraits](#)< Givaro::Montgomery< T > >
- struct [FieldTraits](#)< Field >

*FieldTrait.*

- struct [FieldTraits](#)< Givaro::ZRing< ReclInt::ruint< K > > >
- struct [FieldTraits](#)< Givaro::Modular< Element > >
- struct [FieldTraits](#)< Givaro::ModularBalanced< Element > >
- struct [FieldTraits](#)< Givaro::ZRing< double > >
- struct [FieldTraits](#)< Givaro::ZRing< float > >
- struct [FieldTraits](#)< Givaro::ZRing< int16\_t > >
- struct [FieldTraits](#)< Givaro::ZRing< uint16\_t > >
- struct [FieldTraits](#)< Givaro::ZRing< int32\_t > >
- struct [FieldTraits](#)< Givaro::ZRing< uint32\_t > >
- struct [FieldTraits](#)< Givaro::ZRing< int64\_t > >
- struct [FieldTraits](#)< Givaro::ZRing< uint64\_t > >
- struct [FieldTraits](#)< Givaro::ZRing< Givaro::Integer > >
- struct [FieldTraits](#)< FFPACK::RNSInteger< T > >
- struct [FieldTraits](#)< FFPACK::RNSIntegerMod< T > >
- struct [associatedDelayedField](#)< Field >
- struct [associatedDelayedField](#)< const Givaro::Modular< T, X > >
- struct [associatedDelayedField](#)< const Givaro::ModularBalanced< T > >
- struct [associatedDelayedField](#)< const Givaro::ZRing< T > >
- struct [associatedDelayedField](#)< const FFPACK::RNSIntegerMod< RNS > >

## Namespaces

- namespace [ReclInt](#)
- namespace [Givaro](#)
- namespace [FFPACK](#)

*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

- namespace [FFLAS](#)
- namespace [FFLAS::FieldCategories](#)

*Traits and categories will need to be placed in a proper file later.*

- namespace [FFLAS::ModeCategories](#)

*Specifies the mode of action for an algorithm w.r.t.*

- namespace [FFLAS::ElementCategories](#)



## Functions

- template<class [Field](#) , class enable = void>  
Field::Residu\_t [maxCardinality](#) ()
- template<> uint64\_t [maxCardinality](#)< Givaro::Modular< int64\_t > > ()
- template<> uint32\_t [maxCardinality](#)< Givaro::Modular< int32\_t > > ()
- template<class [Field](#) >  
Field::Residu\_t [minCardinality](#) ()

### 13.208.1 Detailed Description

Field Traits.

## 13.209 field.doxy File Reference

### 13.210 rns-double-elt.h File Reference

rns elt structure with double support

```
#include "fflas-ffpack/utils/fflas_memory.h"
#include "fflas-ffpack/utils/cast.h"
```

## Data Structures

- struct [rns\\_double\\_elt](#)
- struct [rns\\_double\\_elt\\_ptr](#)
- struct [rns\\_double\\_elt\\_cstptr](#)

## Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

## Functions

- template<> [rns\\_double\\_elt\\_ptr](#) fflas\_const\_cast ([rns\\_double\\_elt\\_cstptr](#) x)
- template<> [rns\\_double\\_elt\\_cstptr](#) fflas\_const\_cast ([rns\\_double\\_elt\\_ptr](#) x)

### 13.210.1 Detailed Description

rns elt structure with double support

### 13.211 rns-double-recint.inl File Reference

```
#include "fflas-ffpack/fflas/fflas_freduce.h"
```

## Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

## Macros

- #define [\\_\\_FFLASFFPACK\\_field\\_rns\\_double\\_recint\\_INL](#)

### 13.211.1 Macro Definition Documentation

#### 13.211.1.1 \_\_FFLASFFPACK\_field\_rns\_double\_recint\_INL

```
#define __FFLASFFPACK_field_rns_double_recint_INL
```

## 13.212 rns-double.h File Reference

rns structure with double support

```
#include <iterator>
#include <vector>
#include <givaro/modular-floating.h>
#include <givaro/givinteger.h>
#include <givaro/givintprime.h>
#include "givaro/modular-extended.h"
#include <recint/ruint.h>
#include "fflas-ffpack/config-blas.h"
#include "fflas-ffpack/utils/fflas_memory.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include "fflas-ffpack/field/rns-double-elt.h"
#include "rns-double.inl"
#include "rns-double-recint.inl"
```

### Data Structures

- struct [rns\\_double](#)
- struct [rns\\_double\\_extended](#)
- class [rnsRandIter](#)< [RNS](#) >

### Namespaces

- namespace [FFPACK](#)  
*Finite Field PACK Set of elimination based routines for dense linear algebra.*
- namespace [FFLAS](#)

### Macros

- #define [ROUND\\_DOWN](#)(x, s)

### Functions

- template<> void [fflas\\_delete](#) ([FFPACK::rns\\_double\\_elt\\_ptr](#) A)
- template<> void [fflas\\_delete](#) ([FFPACK::rns\\_double\\_elt\\_cstptr](#) A)

### 13.212.1 Detailed Description

rns structure with double support

### 13.212.2 Macro Definition Documentation

#### 13.212.2.1 ROUND\_DOWN

```
#define ROUND_DOWN (
    x,
    s)
```

**Value:**

```
((x) & ~( (s)-1))
```

## 13.213 rns-double.inl File Reference

```
#include "fflas-ffpack/fflas/fflas_freduce.h"
```

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [\\_\\_FFLASFFPACK\\_field\\_rns\\_double\\_INL](#)

### 13.213.1 Macro Definition Documentation

#### 13.213.1.1 [\\_\\_FFLASFFPACK\\_field\\_rns\\_double\\_INL](#)

```
#define __FFLASFFPACK_field_rns_double_INL
```

## 13.214 rns-integer-mod.h File Reference

representation of  $\mathbb{Z}/p\mathbb{Z}$  using RNS representation (note: fixed precision)

```
#include <vector>
#include <cmath>
#include <recint/recint.h>
#include <givaro/modular-integer.h>
#include <givaro/givinteger.h>
#include <givaro/udl.h>
#include "givaro/modular-extended.h"
#include "fflas-ffpack/field/rns-double.h"
#include "fflas-ffpack/field/rns-integer.h"
#include "fflas-ffpack/fflas/fflas_level1.inl"
#include "fflas-ffpack/fflas/fflas_level2.inl"
#include "fflas-ffpack/fflas/fflas_level3.inl"
#include "fflas-ffpack/fflas/fflas_enum.h"
#include "fflas-ffpack/fflas/fflas_fscal_mp.inl"
```

### Data Structures

- class [RNSIntegerMod< RNS >](#)
- class [RNSIntegerMod< RNS >::RandIter](#)

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*
- namespace [FFLAS](#)

### Functions

- template<> [FFPACK::rns\\_double\\_elt\\_ptr fflas\\_new](#) (const [FFPACK::RNSIntegerMod< FFPACK::rns\\_double >](#) &F, const size\_t m, const Alignment align)
- template<> [FFPACK::rns\\_double\\_elt\\_ptr fflas\\_new](#) (const [FFPACK::RNSIntegerMod< FFPACK::rns\\_double >](#) &F, const size\_t m, const size\_t n, const Alignment align)

- template<typename [RNS](#) >  
void [finit\\_rns](#) (const [FFPACK::RNSIntegerMod](#)< [RNS](#) > &F, const size\_t m, const size\_t n, size\_t k, const Givaro::Integer \*B, const size\_t ldb, typename [RNS::Element\\_ptr](#) A)
- template<typename [RNS](#) >  
void [finit\\_trans\\_rns](#) (const [FFPACK::RNSIntegerMod](#)< [RNS](#) > &F, const size\_t m, const size\_t n, size\_t k, const Givaro::Integer \*B, const size\_t ldb, typename [RNS::Element\\_ptr](#) A)
- template<typename [RNS](#) >  
void [fconvert\\_rns](#) (const [FFPACK::RNSIntegerMod](#)< [RNS](#) > &F, const size\_t m, const size\_t n, Givaro::Integer alpha, Givaro::Integer \*B, const size\_t ldb, typename [RNS::ConstElement\\_ptr](#) A)
- template<typename [RNS](#) >  
void [fconvert\\_trans\\_rns](#) (const [FFPACK::RNSIntegerMod](#)< [RNS](#) > &F, const size\_t m, const size\_t n, Givaro::Integer alpha, Givaro::Integer \*B, const size\_t ldb, typename [RNS::ConstElement\\_ptr](#) A)

### 13.214.1 Detailed Description

representation of  $\mathbb{Z}/p\mathbb{Z}$  using RNS representation (note: fixed precision)

## 13.215 rns-integer.h File Reference

representation of  $\mathbb{Z}$  using RNS representation (note: fixed precision)

```
#include <givaro/givinteger.h>
#include "fflas-ffpack/field/rns-double.h"
```

### Data Structures

- class [RNSInteger](#)< [RNS](#) >
- class [RNSInteger](#)< [RNS](#) >::RandIter

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*
- namespace [FFLAS](#)

### Functions

- template<> [FFPACK::rns\\_double\\_elt\\_ptr fflas\\_new](#) (const [FFPACK::RNSInteger](#)< [FFPACK::rns\\_double](#) > &F, const size\_t m, const Alignment align)
- template<> [FFPACK::rns\\_double\\_elt\\_ptr fflas\\_new](#) (const [FFPACK::RNSInteger](#)< [FFPACK::rns\\_double](#) > &F, const size\_t m, const size\_t n, const Alignment align)
- template<typename [RNS](#) >  
void [finit\\_rns](#) (const [FFPACK::RNSInteger](#)< [RNS](#) > &F, const size\_t m, const size\_t n, size\_t k, const Givaro::Integer \*B, const size\_t ldb, typename [FFPACK::RNSInteger](#)< [RNS](#) >::Element\_ptr A)
- template<typename [RNS](#) >  
void [fconvert\\_rns](#) (const [FFPACK::RNSInteger](#)< [RNS](#) > &F, const size\_t m, const size\_t n, Givaro::Integer alpha, Givaro::Integer \*B, const size\_t ldb, typename [FFPACK::RNSInteger](#)< [RNS](#) >::ConstElement\_ptr A)

### 13.215.1 Detailed Description

representation of  $\mathbb{Z}$  using RNS representation (note: fixed precision)

## 13.216 rns.h File Reference

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

## 13.217 rns.inl File Reference

```
#include "rns-double.h"
#include "rns-integer.h"
#include "rns-integer-mod.h"
```

### Macros

- #define [\\_\\_FFLASFFPACK\\_field\\_rns\\_INL](#)

### 13.217.1 Macro Definition Documentation

#### 13.217.1.1 \_\_FFLASFFPACK\_field\_rns\_INL

```
#define __FFLASFFPACK_field_rns_INL
```

## 13.218 interfaces.doxy File Reference

## 13.219 fflas\_c.h File Reference

```
#include <stdbool.h>
#include <stdlib.h>
#include <inttypes.h>
```

### Macros

- #define [FFLAS\\_COMPILED](#)

### Enumerations

- enum [FFLAS\\_C\\_ORDER](#) { [FflasRowMajor](#) = 101 , [FflasColMajor](#) = 102 }  
*Storage by row or col ?*
- enum [FFLAS\\_C\\_TRANSPOSE](#) { [FflasNoTrans](#) = 111 , [FflasTrans](#) = 112 }  
*Is matrix transposed ?*
- enum [FFLAS\\_C\\_UPLO](#) { [FflasUpper](#) = 121 , [FflasLower](#) = 122 }  
*Is triangular matrix's shape upper ?*
- enum [FFLAS\\_C\\_DIAG](#) { [FflasNonUnit](#) = 131 , [FflasUnit](#) = 132 }  
*Is the triangular matrix implicitly unit diagonal ?*
- enum [FFLAS\\_C\\_SIDE](#) { [FflasLeft](#) = 141 , [FflasRight](#) = 142 }  
*On what side ?*
- enum [FFLAS\\_C\\_BASE](#) { [FflasDouble](#) = 151 , [FflasFloat](#) = 152 , [FflasGeneric](#) = 153 }  
*FFLAS\_C\_BASE determines the type of the element representation for Matrix Mult kernel.*

### Functions

- void [freducein\\_1\\_modular\\_double](#) (const double p, const size\_t n, double \*X, const size\_t incX, bool positive)
- void [freduce\\_1\\_modular\\_double](#) (const double F, const size\_t n, const double \*Y, const size\_t incY, double \*X, const size\_t incX, bool positive)
- void [fnegin\\_1\\_modular\\_double](#) (const double F, const size\_t n, double \*X, const size\_t incX, bool positive)
- void [fneg\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*Y, const size\_t incY, double \*X, const size\_t incX, bool positive)
- void [fzero\\_1\\_modular\\_double](#) (const double p, const size\_t n, double \*X, const size\_t incX, bool positive)
- bool [fiszero\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*X, const size\_t incX, bool positive)

- bool [fequal\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*X, const size\_t incX, const double \*Y, const size\_t incY, bool positive)
- void [fassign\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*Y, const size\_t incY, double \*X, const size\_t incX, bool positive)
- void [fscalin\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double alpha, double \*X, const size\_t incX, bool positive)
- void [fscal\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double alpha, const double \*X, const size\_t incX, double \*Y, const size\_t incY, bool positive)
- void [faxpy\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double alpha, const double \*X, const size\_t incX, double \*Y, const size\_t incY, bool positive)
- double [fdot\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*X, const size\_t incX, const double \*Y, const size\_t incY, bool positive)
- void [fswap\\_1\\_modular\\_double](#) (const double p, const size\_t n, double \*X, const size\_t incX, double \*Y, const size\_t incY, bool positive)
- void [fadd\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*A, const size\_t incA, const double \*B, const size\_t incB, double \*C, const size\_t incC, bool positive)
- void [fsub\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*A, const size\_t incA, const double \*B, const size\_t incB, double \*C, const size\_t incC, bool positive)
- void [faddin\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*B, const size\_t incB, double \*C, const size\_t incC, bool positive)
- void [fsubin\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*B, const size\_t incB, double \*C, const size\_t incC, bool positive)
- void [fassign\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*B, const size\_t incB, double \*A, const size\_t incA, bool positive)
- void [fzero\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, double \*A, const size\_t incA, bool positive)
- bool [fequal\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*A, const size\_t incA, const double \*B, const size\_t incB, bool positive)
- bool [fiszero\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*A, const size\_t incA, bool positive)
- void [fidentity\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, double \*A, const size\_t incA, const double d, bool positive)
- void [freducein\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, double \*A, const size\_t incA, bool positive)
- void [freduce\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*B, const size\_t incB, double \*A, const size\_t incA, bool positive)
- void [fnegin\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, double \*A, const size\_t incA, bool positive)
- void [fneg\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*B, const size\_t incB, double \*A, const size\_t incA, bool positive)
- void [fscalin\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double alpha, double \*A, const size\_t incA, bool positive)
- void [fscal\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double alpha, const double \*A, const size\_t incA, double \*B, const size\_t incB, bool positive)
- void [faxpy\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double alpha, const double \*X, const size\_t incX, double \*Y, const size\_t incY, bool positive)
- void [fmove\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, double \*A, const size\_t incA, double \*B, const size\_t incB, bool positive)
- void [fadd\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*A, const size\_t incA, const double \*B, const size\_t incB, double \*C, const size\_t incC, bool positive)
- void [fsub\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*A, const size\_t incA, const double \*B, const size\_t incB, double \*C, const size\_t incC, bool positive)
- void [fsubin\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*B, const size\_t incB, double \*C, const size\_t incC, bool positive)
- void [faddin\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*B, const size\_t incB, double \*C, const size\_t incC, bool positive)

- double \* [fgemv\\_2\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_TRANSPOSE](#) TransA, const size\_t m, const size\_t n, const double alpha, const double \*A, const size\_t ldA, const double \*X, const size\_t incX, const double betA, double \*Y, const size\_t incY, bool positive)
- void [fger\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double alpha, const double \*x, const size\_t incX, const double \*y, const size\_t incY, double \*A, const size\_t ldA, bool positive)
- void [ftrsv\\_2\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const enum [FFLAS\\_C\\_TRANSPOSE](#) TransA, const enum [FFLAS\\_C\\_DIAG](#) Diag, const size\_t n, const double \*A, const size\_t ldA, double \*X, int incX, bool positive)
- void [ftrsm\\_3\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_SIDE](#) Side, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const enum [FFLAS\\_C\\_TRANSPOSE](#) TransA, const enum [FFLAS\\_C\\_DIAG](#) Diag, const size\_t m, const size\_t n, const double alpha, const double \*A, const size\_t ldA, double \*B, const size\_t ldB, bool positive)
- void [ftrmm\\_3\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_SIDE](#) Side, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const enum [FFLAS\\_C\\_TRANSPOSE](#) TransA, const enum [FFLAS\\_C\\_DIAG](#) Diag, const size\_t m, const size\_t n, const double alpha, double \*A, const size\_t ldA, double \*B, const size\_t ldB, bool positive)
- double \* [fgemm\\_3\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_TRANSPOSE](#) tA, const enum [FFLAS\\_C\\_TRANSPOSE](#) tB, const size\_t m, const size\_t n, const size\_t k, const double alpha, const double \*A, const size\_t ldA, const double \*B, const size\_t ldB, const double betA, double \*C, const size\_t ldC, bool positive)
- double \* [fsquare\\_3\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_TRANSPOSE](#) tA, const size\_t n, const double alpha, const double \*A, const size\_t ldA, const double betA, double \*C, const size\_t ldC, bool positive)

## 13.219.1 Macro Definition Documentation

### 13.219.1.1 FFLAS\_COMPILED

```
#define FFLAS_COMPILED
```

## 13.219.2 Enumeration Type Documentation

### 13.219.2.1 FFLAS\_C\_ORDER

```
enum FFLAS\_C\_ORDER
```

Storage by row or col ?

Enumerator

|                               |           |
|-------------------------------|-----------|
| <a href="#">FflasRowMajor</a> | row major |
| <a href="#">FflasColMajor</a> | col major |

### 13.219.2.2 FFLAS\_C\_TRANSPOSE

```
enum FFLAS\_C\_TRANSPOSE
```

Is matrix transposed ?

Enumerator

|                              |                           |
|------------------------------|---------------------------|
| <a href="#">FflasNoTrans</a> | Matrix is not transposed. |
| <a href="#">FflasTrans</a>   | Matrix is transposed.     |

### 13.219.2.3 FFLAS\_C\_UPLO

```
enum FFLAS\_C\_UPLO
```

Is triangular matrix's shape upper ?

## Enumerator

|            |                                                                        |
|------------|------------------------------------------------------------------------|
| FflasUpper | Triangular matrix is Upper triangular (if $i > j$ then $T_{i,j} = 0$ ) |
| FflasLower | Triangular matrix is Lower triangular (if $i < j$ then $T_{i,j} = 0$ ) |

**13.219.2.4 FFLAS\_C\_DIAG**enum [FFLAS\\_C\\_DIAG](#)

Is the triangular matrix implicitly unit diagonal ?

## Enumerator

|              |                                                                   |
|--------------|-------------------------------------------------------------------|
| FflasNonUnit | Triangular matrix has an explicit arbitrary diagonal.             |
| FflasUnit    | Triangular matrix has an implicit unit diagonal ( $T_{i,i} = 1$ ) |

**13.219.2.5 FFLAS\_C\_SIDE**enum [FFLAS\\_C\\_SIDE](#)

On what side ?

## Enumerator

|            |                                 |
|------------|---------------------------------|
| FflasLeft  | Operator applied on the left.   |
| FflasRight | Operator applied on the righth. |

**13.219.2.6 FFLAS\_C\_BASE**enum [FFLAS\\_C\\_BASE](#)

FFLAS\_C\_BASE determines the type of the element representation for Matrix Mult kernel.  
(deprecated, should not be used)

## Enumerator

|              |                                                                            |
|--------------|----------------------------------------------------------------------------|
| FflasDouble  | to use the double precision BLAS                                           |
| FflasFloat   | to use the single precision BLAS                                           |
| FflasGeneric | for any other domain, that can not be converted to floating point integers |

**13.219.3 Function Documentation****13.219.3.1 freducein\_1\_modular\_double()**

```
void freducein_1_modular_double (
    const double p,
    const size_t n,
    double * X,
    const size_t incX,
    bool positive)
```



### 13.219.3.2 freduce\_1\_modular\_double()

```
void freduce_1_modular_double (
    const double F,
    const size_t n,
    const double * Y,
    const size_t incY,
    double * X,
    const size_t incX,
    bool positive)
```

### 13.219.3.3 fnegin\_1\_modular\_double()

```
void fnegin_1_modular_double (
    const double F,
    const size_t n,
    double * X,
    const size_t incX,
    bool positive)
```

### 13.219.3.4 fneg\_1\_modular\_double()

```
void fneg_1_modular_double (
    const double p,
    const size_t n,
    const double * Y,
    const size_t incY,
    double * X,
    const size_t incX,
    bool positive)
```

### 13.219.3.5 fzero\_1\_modular\_double()

```
void fzero_1_modular_double (
    const double p,
    const size_t n,
    double * X,
    const size_t incX,
    bool positive)
```

### 13.219.3.6 fiszero\_1\_modular\_double()

```
bool fiszero_1_modular_double (
    const double p,
    const size_t n,
    const double * X,
    const size_t incX,
    bool positive)
```

### 13.219.3.7 fequal\_1\_modular\_double()

```
bool fequal_1_modular_double (
    const double p,
    const size_t n,
    const double * X,
    const size_t incX,
    const double * Y,
    const size_t incY,
    bool positive)
```

**13.219.3.8 fassign\_1\_modular\_double()**

```
void fassign_1_modular_double (
    const double p,
    const size_t n,
    const double * Y,
    const size_t incY,
    double * X,
    const size_t incX,
    bool positive)
```

**13.219.3.9 fscaln\_1\_modular\_double()**

```
void fscaln_1_modular_double (
    const double p,
    const size_t n,
    const double alpha,
    double * X,
    const size_t incX,
    bool positive)
```

**13.219.3.10 fscal\_1\_modular\_double()**

```
void fscal_1_modular_double (
    const double p,
    const size_t n,
    const double alpha,
    const double * X,
    const size_t incX,
    double * Y,
    const size_t incY,
    bool positive)
```

**13.219.3.11 faxpy\_1\_modular\_double()**

```
void faxpy_1_modular_double (
    const double p,
    const size_t n,
    const double alpha,
    const double * X,
    const size_t incX,
    double * Y,
    const size_t incY,
    bool positive)
```

**13.219.3.12 fdot\_1\_modular\_double()**

```
double fdot_1_modular_double (
    const double p,
    const size_t n,
    const double * X,
    const size_t incX,
    const double * Y,
    const size_t incY,
    bool positive)
```

**13.219.3.13 fswap\_1\_modular\_double()**

```
void fswap_1_modular_double (
    const double p,
```

```
    const size_t n,  
    double * X,  
    const size_t incX,  
    double * Y,  
    const size_t incY,  
    bool positive)
```

#### 13.219.3.14 fadd\_1\_modular\_double()

```
void fadd_1_modular_double (  
    const double p,  
    const size_t n,  
    const double * A,  
    const size_t incA,  
    const double * B,  
    const size_t incB,  
    double * C,  
    const size_t incC,  
    bool positive)
```

#### 13.219.3.15 fsub\_1\_modular\_double()

```
void fsub_1_modular_double (  
    const double p,  
    const size_t n,  
    const double * A,  
    const size_t incA,  
    const double * B,  
    const size_t incB,  
    double * C,  
    const size_t incC,  
    bool positive)
```

#### 13.219.3.16 faddin\_1\_modular\_double()

```
void faddin_1_modular_double (  
    const double p,  
    const size_t n,  
    const double * B,  
    const size_t incB,  
    double * C,  
    const size_t incC,  
    bool positive)
```

#### 13.219.3.17 fsubin\_1\_modular\_double()

```
void fsubin_1_modular_double (  
    const double p,  
    const size_t n,  
    const double * B,  
    const size_t incB,  
    double * C,  
    const size_t incC,  
    bool positive)
```

#### 13.219.3.18 fassign\_2\_modular\_double()

```
void fassign_2_modular_double (  
    const double p,
```

```

    const size_t m,
    const size_t n,
    const double * B,
    const size_t ldB,
    double * A,
    const size_t ldA,
    bool positive)

```

#### 13.219.3.19 fzero\_2\_modular\_double()

```

void fzero_2_modular_double (
    const double p,
    const size_t m,
    const size_t n,
    double * A,
    const size_t ldA,
    bool positive)

```

#### 13.219.3.20 fequal\_2\_modular\_double()

```

bool fequal_2_modular_double (
    const double p,
    const size_t m,
    const size_t n,
    const double * A,
    const size_t ldA,
    const double * B,
    const size_t ldB,
    bool positive)

```

#### 13.219.3.21 fiszero\_2\_modular\_double()

```

bool fiszero_2_modular_double (
    const double p,
    const size_t m,
    const size_t n,
    const double * A,
    const size_t ldA,
    bool positive)

```

#### 13.219.3.22 fidentity\_2\_modular\_double()

```

void fidentity_2_modular_double (
    const double p,
    const size_t m,
    const size_t n,
    double * A,
    const size_t ldA,
    const double d,
    bool positive)

```

#### 13.219.3.23 freducein\_2\_modular\_double()

```

void freducein_2_modular_double (
    const double p,
    const size_t m,
    const size_t n,
    double * A,

```

```
    const size_t ldA,  
    bool positive)
```

#### 13.219.3.24 freduce\_2\_modular\_double()

```
void freduce_2_modular_double (  
    const double p,  
    const size_t m,  
    const size_t n,  
    const double * B,  
    const size_t ldB,  
    double * A,  
    const size_t ldA,  
    bool positive)
```

#### 13.219.3.25 fnegin\_2\_modular\_double()

```
void fnegin_2_modular_double (  
    const double p,  
    const size_t m,  
    const size_t n,  
    double * A,  
    const size_t ldA,  
    bool positive)
```

#### 13.219.3.26 fneg\_2\_modular\_double()

```
void fneg_2_modular_double (  
    const double p,  
    const size_t m,  
    const size_t n,  
    const double * B,  
    const size_t ldB,  
    double * A,  
    const size_t ldA,  
    bool positive)
```

#### 13.219.3.27 fscaln\_2\_modular\_double()

```
void fscaln_2_modular_double (  
    const double p,  
    const size_t m,  
    const size_t n,  
    const double alpha,  
    double * A,  
    const size_t ldA,  
    bool positive)
```

#### 13.219.3.28 fscal\_2\_modular\_double()

```
void fscal_2_modular_double (  
    const double p,  
    const size_t m,  
    const size_t n,  
    const double alpha,  
    const double * A,  
    const size_t ldA,  
    double * B,
```

```
    const size_t ldB,  
    bool positive)
```

#### 13.219.3.29 faxpy\_2\_modular\_double()

```
void faxpy_2_modular_double (  
    const double p,  
    const size_t m,  
    const size_t n,  
    const double alpha,  
    const double * X,  
    const size_t ldX,  
    double * Y,  
    const size_t ldY,  
    bool positive)
```

#### 13.219.3.30 fmove\_2\_modular\_double()

```
void fmove_2_modular_double (  
    const double p,  
    const size_t m,  
    const size_t n,  
    double * A,  
    const size_t ldA,  
    double * B,  
    const size_t ldB,  
    bool positive)
```

#### 13.219.3.31 fadd\_2\_modular\_double()

```
void fadd_2_modular_double (  
    const double p,  
    const size_t m,  
    const size_t n,  
    const double * A,  
    const size_t ldA,  
    const double * B,  
    const size_t ldB,  
    double * C,  
    const size_t ldC,  
    bool positive)
```

#### 13.219.3.32 fsub\_2\_modular\_double()

```
void fsub_2_modular_double (  
    const double p,  
    const size_t m,  
    const size_t n,  
    const double * A,  
    const size_t ldA,  
    const double * B,  
    const size_t ldB,  
    double * C,  
    const size_t ldC,  
    bool positive)
```

#### 13.219.3.33 fsubin\_2\_modular\_double()

```
void fsubin_2_modular_double (  

```

```
const double p,  
const size_t m,  
const size_t n,  
const double * B,  
const size_t ldB,  
double * C,  
const size_t ldC,  
bool positive)
```

#### 13.219.3.34 faddin\_2\_modular\_double()

```
void faddin_2_modular_double (  
    const double p,  
    const size_t m,  
    const size_t n,  
    const double * B,  
    const size_t ldB,  
    double * C,  
    const size_t ldC,  
    bool positive)
```

#### 13.219.3.35 fgemv\_2\_modular\_double()

```
double * fgemv_2_modular_double (  
    const double p,  
    const enum FFLAS_C_TRANSPOSE TransA,  
    const size_t m,  
    const size_t n,  
    const double alpha,  
    const double * A,  
    const size_t ldA,  
    const double * X,  
    const size_t incX,  
    const double betA,  
    double * Y,  
    const size_t incY,  
    bool positive)
```

#### 13.219.3.36 fger\_2\_modular\_double()

```
void fger_2_modular_double (  
    const double p,  
    const size_t m,  
    const size_t n,  
    const double alpha,  
    const double * x,  
    const size_t incX,  
    const double * y,  
    const size_t incY,  
    double * A,  
    const size_t ldA,  
    bool positive)
```

#### 13.219.3.37 ftrsv\_2\_modular\_double()

```
void ftrsv_2_modular_double (  
    const double p,  
    const enum FFLAS_C_UPLO Uplo,  
    const enum FFLAS_C_TRANSPOSE TransA,
```

```

    const enum FFLAS_C_DIAG Diag,
    const size_t n,
    const double * A,
    const size_t ldA,
    double * X,
    int incX,
    bool positive)

```

#### 13.219.3.38 ftrsm\_3\_modular\_double()

```

void ftrsm_3_modular_double (
    const double p,
    const enum FFLAS_C_SIDE Side,
    const enum FFLAS_C_UPLO Uplo,
    const enum FFLAS_C_TRANSPOSE TransA,
    const enum FFLAS_C_DIAG Diag,
    const size_t m,
    const size_t n,
    const double alpha,
    const double * A,
    const size_t ldA,
    double * B,
    const size_t ldB,
    bool positive)

```

#### 13.219.3.39 ftrmm\_3\_modular\_double()

```

void ftrmm_3_modular_double (
    const double p,
    const enum FFLAS_C_SIDE Side,
    const enum FFLAS_C_UPLO Uplo,
    const enum FFLAS_C_TRANSPOSE TransA,
    const enum FFLAS_C_DIAG Diag,
    const size_t m,
    const size_t n,
    const double alpha,
    double * A,
    const size_t ldA,
    double * B,
    const size_t ldB,
    bool positive)

```

#### 13.219.3.40 fgemm\_3\_modular\_double()

```

double * fgemm_3_modular_double (
    const double p,
    const enum FFLAS_C_TRANSPOSE tA,
    const enum FFLAS_C_TRANSPOSE tB,
    const size_t m,
    const size_t n,
    const size_t k,
    const double alpha,
    const double * A,
    const size_t ldA,
    const double * B,
    const size_t ldB,
    const double betaA,
    double * C,

```



```
const size_t ldC,
bool positive)
```

### 13.219.3.41 fsquare\_3\_modular\_double()

```
double * fsquare_3_modular_double (
    const double p,
    const enum FFLAS_C_TRANSPOSE tA,
    const size_t n,
    const double alpha,
    const double * A,
    const size_t ldA,
    const double betA,
    double * C,
    const size_t ldC,
    bool positive)
```

## 13.220 fflas\_L1\_inst.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "givaro/modular.h"
#include "givaro/modular-balanced.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/fflas/fflas_helpers.inl"
#include "fflas_L1_inst_implement.inl"
```

### Macros

- #define `__FFLAS_L1_INST_C`
- #define `INST_OR_DECL`
- #define `FFLAS_FIELD Givaro::ModularBalanced`
- #define `FFLAS_ELT double`
- #define `FFLAS_ELT float`
- #define `FFLAS_ELT int64_t`
- #define `FFLAS_FIELD Givaro::Modular`
- #define `FFLAS_ELT double`
- #define `FFLAS_ELT float`
- #define `FFLAS_ELT int64_t`

### 13.220.1 Macro Definition Documentation

#### 13.220.1.1 \_\_FFLAS\_L1\_INST\_C

```
#define __FFLAS_L1_INST_C
```

#### 13.220.1.2 INST\_OR\_DECL

```
#define INST_OR_DECL
```

#### 13.220.1.3 FFLAS\_FIELD [1/2]

```
#define FFLAS_FIELD Givaro::ModularBalanced
```

#### 13.220.1.4 FFLAS\_ELT [1/6]

```
#define FFLAS_ELT double
```

**13.220.1.5 FFLAS\_ELT [2/6]**

```
#define FFLAS_ELT float
```

**13.220.1.6 FFLAS\_ELT [3/6]**

```
#define FFLAS_ELT int64_t
```

**13.220.1.7 FFLAS\_FIELD [2/2]**

```
#define FFLAS_FIELD Givaro::Modular
```

**13.220.1.8 FFLAS\_ELT [4/6]**

```
#define FFLAS_ELT double
```

**13.220.1.9 FFLAS\_ELT [5/6]**

```
#define FFLAS_ELT float
```

**13.220.1.10 FFLAS\_ELT [6/6]**

```
#define FFLAS_ELT int64_t
```

**13.221 fflas\_L1\_inst.h File Reference**

```
#include "givaro/modular.h"
#include "givaro/modular-balanced.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/fflas/fflas_helpers.inl"
#include "fflas_L1_inst_implem.inl"
```

**Macros**

- `#define INST_OR_DECL <>`
- `#define FFLAS_FIELD Givaro::ModularBalanced`
- `#define FFLAS_ELT double`
- `#define FFLAS_ELT float`
- `#define FFLAS_ELT int64_t`
- `#define FFLAS_FIELD Givaro::Modular`
- `#define FFLAS_ELT double`
- `#define FFLAS_ELT float`
- `#define FFLAS_ELT int64_t`

**13.221.1 Macro Definition Documentation****13.221.1.1 INST\_OR\_DECL**

```
#define INST_OR_DECL <>
```

**13.221.1.2 FFLAS\_FIELD [1/2]**

```
#define FFLAS_FIELD Givaro::ModularBalanced
```

**13.221.1.3 FFLAS\_ELT [1/6]**

```
#define FFLAS_ELT double
```

**13.221.1.4 FFLAS\_ELT [2/6]**

```
#define FFLAS_ELT float
```

**13.221.1.5 FFLAS\_ELT [3/6]**

```
#define FFLAS_ELT int64_t
```

**13.221.1.6 FFLAS\_FIELD [2/2]**

```
#define FFLAS_FIELD Givaro::Modular
```

**13.221.1.7 FFLAS\_ELT [4/6]**

```
#define FFLAS_ELT double
```

**13.221.1.8 FFLAS\_ELT [5/6]**

```
#define FFLAS_ELT float
```

**13.221.1.9 FFLAS\_ELT [6/6]**

```
#define FFLAS_ELT int64_t
```

**13.222 fflas\_L1\_inst\_implem.inl File Reference****Namespaces**

- namespace [FFLAS](#)

**Functions**

- template [INST\\_OR\\_DECL](#) void [freduce](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const size\_t n, [FFLAS\\_ELT](#) \*X, const size\_t incX)  

$$\text{freduce } x \leftarrow x \bmod F.$$
- template [INST\\_OR\\_DECL](#) void [freduce](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const size\_t n, const [FFLAS\\_ELT](#) \*Y, const size\_t incY, [FFLAS\\_ELT](#) \*X, const size\_t incX)  

$$\text{freduce } x \leftarrow y \bmod F.$$
- template [INST\\_OR\\_DECL](#) void [finit](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const size\_t n, const [FFLAS\\_ELT](#) \*Y, const size\_t incY, [FFLAS\\_ELT](#) \*X, const size\_t incX)  

$$\text{finit } x \leftarrow y \bmod F.$$
- template [INST\\_OR\\_DECL](#) void [fconvert](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const size\_t n, [FFLAS\\_ELT](#) \*X, const size\_t incX, const [FFLAS\\_ELT](#) \*Y, const size\_t incY)  

$$\text{fconvert } x \leftarrow y \bmod F.$$
- template [INST\\_OR\\_DECL](#) void [fnegin](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const size\_t n, [FFLAS\\_ELT](#) \*X, const size\_t incX)  

$$\text{fnegin } x \leftarrow -x.$$
- template [INST\\_OR\\_DECL](#) void [fneg](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const size\_t n, const [FFLAS\\_ELT](#) \*Y, const size\_t incY, [FFLAS\\_ELT](#) \*X, const size\_t incX)  

$$\text{fneg } x \leftarrow -y.$$
- template [INST\\_OR\\_DECL](#) void [fzero](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const size\_t n, [FFLAS\\_ELT](#) \*X, const size\_t incX)  

$$\text{fzero} : A \leftarrow 0.$$
- template [INST\\_OR\\_DECL](#) bool [fiszero](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const size\_t n, const [FFLAS\\_ELT](#) \*X, const size\_t incX)  

$$\text{fiszero} : \text{test } X = 0.$$

- template `INST_OR_DECL` bool `fequal` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` n, const `FFLAS_ELT` \*X, const `size_t` incX, const `FFLAS_ELT` \*Y, const `size_t` incY)  

$$fequal : test\ X = Y.$$
- template `INST_OR_DECL` void `fassign` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` N, const `FFLAS_ELT` \*Y, const `size_t` incY, `FFLAS_ELT` \*X, const `size_t` incX)  

$$fassign : x \leftarrow y.$$
- template `INST_OR_DECL` void `fscaln` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` n, const `FFLAS_ELT` alpha, `FFLAS_ELT` \*X, const `size_t` incX)  

$$fscaln\ x \leftarrow \alpha \cdot x.$$
- template `INST_OR_DECL` void `fscal` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` n, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*X, const `size_t` incX, `FFLAS_ELT` \*Y, const `size_t` incY)  

$$fscal\ y \leftarrow \alpha \cdot x.$$
- template `INST_OR_DECL` void `faxpy` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` N, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*X, const `size_t` incX, `FFLAS_ELT` \*Y, const `size_t` incY)  

$$faxpy : y \leftarrow \alpha \cdot x + y.$$
- template `INST_OR_DECL` `FFLAS_ELT` `fdot` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` N, const `FFLAS_ELT` \*X, const `size_t` incX, const `FFLAS_ELT` \*Y, const `size_t` incY)  

$$faxpby : y \leftarrow \alpha \cdot x + \beta \cdot y.$$
- template `INST_OR_DECL` void `fswap` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` N, `FFLAS_ELT` \*X, const `size_t` incX, `FFLAS_ELT` \*Y, const `size_t` incY)  

$$fswap : X \leftrightarrow Y.$$
- template `INST_OR_DECL` void `fadd` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` N, const `FFLAS_ELT` \*A, const `size_t` inca, const `FFLAS_ELT` \*B, const `size_t` incb, `FFLAS_ELT` \*C, const `size_t` incc)
- template `INST_OR_DECL` void `fsub` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` N, const `FFLAS_ELT` \*A, const `size_t` inca, const `FFLAS_ELT` \*B, const `size_t` incb, `FFLAS_ELT` \*C, const `size_t` incc)
- template `INST_OR_DECL` void `faddin` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` N, const `FFLAS_ELT` \*B, const `size_t` incb, `FFLAS_ELT` \*C, const `size_t` incc)
- template `INST_OR_DECL` void `fadd` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` N, const `FFLAS_ELT` \*A, const `size_t` inca, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*B, const `size_t` incb, `FFLAS_ELT` \*C, const `size_t` incc)

### 13.223 fflas\_L2\_inst.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "givaro/modular.h"
#include "givaro/modular-balanced.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/fflas/fflas_helpers.inl"
#include "fflas_L2_inst_implem.inl"
```

#### Macros

- #define `__FFLAS_L2_INST_C`
- #define `INST_OR_DECL`
- #define `FFLAS_FIELD` Givaro::ModularBalanced
- #define `FFLAS_ELT` double
- #define `FFLAS_ELT` float
- #define `FFLAS_ELT` int64\_t
- #define `FFLAS_FIELD` Givaro::Modular
- #define `FFLAS_ELT` double
- #define `FFLAS_ELT` float
- #define `FFLAS_ELT` int64\_t

## 13.223.1 Macro Definition Documentation

### 13.223.1.1 \_\_FFLAS\_L2\_INST\_C

```
#define __FFLAS_L2_INST_C
```

### 13.223.1.2 INST\_OR\_DECL

```
#define INST_OR_DECL
```

### 13.223.1.3 FFLAS\_FIELD [1/2]

```
#define FFLAS_FIELD Givaro::ModularBalanced
```

### 13.223.1.4 FFLAS\_ELT [1/6]

```
#define FFLAS_ELT double
```

### 13.223.1.5 FFLAS\_ELT [2/6]

```
#define FFLAS_ELT float
```

### 13.223.1.6 FFLAS\_ELT [3/6]

```
#define FFLAS_ELT int64_t
```

### 13.223.1.7 FFLAS\_FIELD [2/2]

```
#define FFLAS_FIELD Givaro::Modular
```

### 13.223.1.8 FFLAS\_ELT [4/6]

```
#define FFLAS_ELT double
```

### 13.223.1.9 FFLAS\_ELT [5/6]

```
#define FFLAS_ELT float
```

### 13.223.1.10 FFLAS\_ELT [6/6]

```
#define FFLAS_ELT int64_t
```

## 13.224 fflas\_L2\_inst.h File Reference

```
#include "givaro/modular.h"
#include "givaro/modular-balanced.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/fflas/fflas_helpers.inl"
#include "fflas_L2_inst_implem.inl"
```

### Macros

- #define [INST\\_OR\\_DECL](#) <>
- #define [FFLAS\\_FIELD](#) [Givaro::ModularBalanced](#)
- #define [FFLAS\\_ELT](#) double
- #define [FFLAS\\_ELT](#) float
- #define [FFLAS\\_ELT](#) int64\_t
- #define [FFLAS\\_FIELD](#) [Givaro::Modular](#)

- `#define FFLAS_ELT double`
- `#define FFLAS_ELT float`
- `#define FFLAS_ELT int64_t`

## 13.224.1 Macro Definition Documentation

### 13.224.1.1 INST\_OR\_DECL

```
#define INST_OR_DECL <>
```

### 13.224.1.2 FFLAS\_FIELD [1/2]

```
#define FFLAS_FIELD Givaro::ModularBalanced
```

### 13.224.1.3 FFLAS\_ELT [1/6]

```
#define FFLAS_ELT double
```

### 13.224.1.4 FFLAS\_ELT [2/6]

```
#define FFLAS_ELT float
```

### 13.224.1.5 FFLAS\_ELT [3/6]

```
#define FFLAS_ELT int64_t
```

### 13.224.1.6 FFLAS\_FIELD [2/2]

```
#define FFLAS_FIELD Givaro::Modular
```

### 13.224.1.7 FFLAS\_ELT [4/6]

```
#define FFLAS_ELT double
```

### 13.224.1.8 FFLAS\_ELT [5/6]

```
#define FFLAS_ELT float
```

### 13.224.1.9 FFLAS\_ELT [6/6]

```
#define FFLAS_ELT int64_t
```

## 13.225 fflas\_L2\_inst\_implem.inl File Reference

### Namespaces

- namespace `FFLAS`

### Functions

- template `INST_OR_DECL` void `fassign` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` m, const `size_t` n, const `FFLAS_ELT` \*B, const `size_t` ldb, `FFLAS_ELT` \*A, const `size_t` lda)  
 $fassign : A \leftarrow B.$
- template `INST_OR_DECL` void `fzero` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` m, const `size_t` n, `FFLAS_ELT` \*A, const `size_t` lda)  
 $fzero : A \leftarrow 0.$
- template `INST_OR_DECL` bool `fequal` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` m, const `size_t` n, const `FFLAS_ELT` \*A, const `size_t` lda, const `FFLAS_ELT` \*B, const `size_t` ldb)  
 $fequal : test A = B.$

- template `INST_OR_DECL` bool `fiszero` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, const `FFLAS_ELT` \*A, const size\_t lda)  

$$\text{fiszero} : \text{test } A = 0.$$
- template `INST_OR_DECL` void `fidentity` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, `FFLAS_ELT` \*A, const size\_t lda, const `FFLAS_ELT` &d)  

$$\text{creates a diagonal matrix}$$
- template `INST_OR_DECL` void `fidentity` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, `FFLAS_ELT` \*A, const size\_t lda)  

$$\text{creates a diagonal matrix}$$
- template `INST_OR_DECL` void `freduce` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, `FFLAS_ELT` \*A, const size\_t lda)  

$$\text{freduce } A \leftarrow A \bmod F.$$
- template `INST_OR_DECL` void `freduce` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*A, const size\_t lda)  

$$\text{freduce } A \leftarrow B \bmod F.$$
- template `INST_OR_DECL` void `finit` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*A, const size\_t lda)  

$$\text{finit } A \leftarrow B \bmod F.$$
- template `INST_OR_DECL` void `fnegin` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, `FFLAS_ELT` \*A, const size\_t lda)  

$$\text{fnegin } A \leftarrow -A.$$
- template `INST_OR_DECL` void `fneg` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*A, const size\_t lda)  

$$\text{fneg } A \leftarrow -B.$$
- template `INST_OR_DECL` void `fscaln` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, const `FFLAS_ELT` alpha, `FFLAS_ELT` \*A, const size\_t lda)  

$$\text{fscaln } A \leftarrow a \cdot A.$$
- template `INST_OR_DECL` void `fscal` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*A, const size\_t lda, `FFLAS_ELT` \*B, const size\_t ldb)  

$$\text{fscal } B \leftarrow a \cdot A.$$
- template `INST_OR_DECL` void `faxpy` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*X, const size\_t ldx, `FFLAS_ELT` \*Y, const size\_t ldy)  

$$\text{faxpy} : y \leftarrow \alpha \cdot x + y.$$
- template `INST_OR_DECL` void `fmove` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, `FFLAS_ELT` \*A, const size\_t lda, `FFLAS_ELT` \*B, const size\_t ldb)  

$$\text{faxpby} : y \leftarrow \alpha \cdot x + \beta \cdot y.$$
- template `INST_OR_DECL` void `fadd` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, const `FFLAS_ELT` \*A, const size\_t lda, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*C, const size\_t ldc)  

$$\text{fadd} : \text{matrix addition.}$$
- template `INST_OR_DECL` void `fsub` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, const `FFLAS_ELT` \*A, const size\_t lda, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*C, const size\_t ldc)  

$$\text{fsub} : \text{matrix subtraction.}$$
- template `INST_OR_DECL` void `fsubin` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*C, const size\_t ldc)  

$$\text{fsubin } C = C - B$$
- template `INST_OR_DECL` void `fadd` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, const `FFLAS_ELT` \*A, const size\_t lda, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*C, const size\_t ldc)  

$$\text{fadd} : \text{matrix addition with scaling.}$$
- template `INST_OR_DECL` void `faddin` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*C, const size\_t ldc)

*faddin*

- template `INST_OR_DECL FFLAS_ELT * fgemv` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS_TRANSPOSE` TransA, const `size_t` M, const `size_t` N, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*A, const `size_t` lda, const `FFLAS_ELT` \*X, const `size_t` incX, const `FFLAS_ELT` beta, `FFLAS_ELT` \*Y, const `size_t` incY)

*finite prime FFLAS\_FIELD<FFLAS\_ELT> GEneral Matrix Vector multiplication.*

- template `INST_OR_DECL void fger` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` M, const `size_t` N, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*x, const `size_t` incx, const `FFLAS_ELT` \*y, const `size_t` incy, `FFLAS_ELT` \*A, const `size_t` lda)

*fger: rank one update of a general matrix*

- template `INST_OR_DECL void ftrsv` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS_UPLO` Uplo, const `FFLAS_TRANSPOSE` TransA, const `FFLAS_DIAG` Diag, const `size_t` N, const `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_ELT` \*X, int incX)

*ftrsv: TRIangular System solve with Vector Computes  $X \leftarrow \text{op}(A^{-1})X$*

## 13.226 fflas\_L3\_inst.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "givaro/modular.h"
#include "givaro/modular-balanced.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/fflas/fflas_helpers.inl"
#include "fflas_L3_inst_implem.inl"
```

### Macros

- #define `__FFLAS_L3_INST_C`
- #define `INST_OR_DECL`
- #define `FFLAS_FIELD` `Givaro::ModularBalanced`
- #define `FFLAS_ELT` `double`
- #define `FFLAS_ELT` `float`
- #define `FFLAS_ELT` `int64_t`
- #define `FFLAS_FIELD` `Givaro::Modular`
- #define `FFLAS_ELT` `double`
- #define `FFLAS_ELT` `float`
- #define `FFLAS_ELT` `int64_t`

### 13.226.1 Macro Definition Documentation

#### 13.226.1.1 \_\_FFLAS\_L3\_INST\_C

```
#define __FFLAS_L3_INST_C
```

#### 13.226.1.2 INST\_OR\_DECL

```
#define INST_OR_DECL
```

#### 13.226.1.3 FFLAS\_FIELD [1/2]

```
#define FFLAS_FIELD Givaro::ModularBalanced
```

#### 13.226.1.4 FFLAS\_ELT [1/6]

```
#define FFLAS_ELT double
```



**13.226.1.5 FFLAS\_ELT [2/6]**

```
#define FFLAS_ELT float
```

**13.226.1.6 FFLAS\_ELT [3/6]**

```
#define FFLAS_ELT int64_t
```

**13.226.1.7 FFLAS\_FIELD [2/2]**

```
#define FFLAS_FIELD Givaro::Modular
```

**13.226.1.8 FFLAS\_ELT [4/6]**

```
#define FFLAS_ELT double
```

**13.226.1.9 FFLAS\_ELT [5/6]**

```
#define FFLAS_ELT float
```

**13.226.1.10 FFLAS\_ELT [6/6]**

```
#define FFLAS_ELT int64_t
```

**13.227 fflas\_L3\_inst.h File Reference**

```
#include "givaro/modular.h"
#include "givaro/modular-balanced.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/fflas/fflas_helpers.inl"
#include "fflas_L3_inst_implem.inl"
```

**Macros**

- `#define INST_OR_DECL <>`
- `#define FFLAS_FIELD Givaro::ModularBalanced`
- `#define FFLAS_ELT double`
- `#define FFLAS_ELT float`
- `#define FFLAS_ELT int64_t`
- `#define FFLAS_FIELD Givaro::Modular`
- `#define FFLAS_ELT double`
- `#define FFLAS_ELT float`
- `#define FFLAS_ELT int64_t`

**13.227.1 Macro Definition Documentation****13.227.1.1 INST\_OR\_DECL**

```
#define INST_OR_DECL <>
```

**13.227.1.2 FFLAS\_FIELD [1/2]**

```
#define FFLAS_FIELD Givaro::ModularBalanced
```

**13.227.1.3 FFLAS\_ELT [1/6]**

```
#define FFLAS_ELT double
```

**13.227.1.4 FFLAS\_ELT [2/6]**

```
#define FFLAS_ELT float
```

**13.227.1.5 FFLAS\_ELT [3/6]**

```
#define FFLAS_ELT int64_t
```

**13.227.1.6 FFLAS\_FIELD [2/2]**

```
#define FFLAS_FIELD Givaro::Modular
```

**13.227.1.7 FFLAS\_ELT [4/6]**

```
#define FFLAS_ELT double
```

**13.227.1.8 FFLAS\_ELT [5/6]**

```
#define FFLAS_ELT float
```

**13.227.1.9 FFLAS\_ELT [6/6]**

```
#define FFLAS_ELT int64_t
```

**13.228 fflas\_L3\_inst\_implem.inl File Reference****Namespaces**

- namespace [FFLAS](#)

**Macros**

- #define [\\_\\_FFLAS\\_TRSM\\_READONLY](#)

**Functions**

- template [INST\\_OR\\_DECL](#) void [ftrsm](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const [FFLAS\\_SIDE](#) Side, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, const [FFLAS\\_ELT](#) alpha, const [FFLAS\\_ELT](#) \*A, const size\_t lda, [FFLAS\\_ELT](#) \*B, const size\_t ldb)  
*ftrsm: **TR**iangular **S**ystem solve with **M**atrix.*
- template [INST\\_OR\\_DECL](#) void [ftrmm](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const [FFLAS\\_SIDE](#) Side, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, const [FFLAS\\_ELT](#) alpha, const [FFLAS\\_ELT](#) \*A, const size\_t lda, [FFLAS\\_ELT](#) \*B, const size\_t ldb)  
*ftrmm: **TR**iangular **M**atrix **M**ultiply.*
- template [INST\\_OR\\_DECL](#) [FFLAS\\_ELT](#) \* [fgemm](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t m, const size\_t n, const size\_t k, const [FFLAS\\_ELT](#) alpha, const [FFLAS\\_ELT](#) \*A, const size\_t lda, const [FFLAS\\_ELT](#) \*B, const size\_t ldb, const [FFLAS\\_ELT](#) beta, [FFLAS\\_ELT](#) \*C, const size\_t ldc)  
*fgemm: **F**ield **GE**neral **M**atrix **M**ultiply.*
- template [INST\\_OR\\_DECL](#) [FFLAS\\_ELT](#) \* [fgemm](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t m, const size\_t n, const size\_t k, const [FFLAS\\_ELT](#) alpha, const [FFLAS\\_ELT](#) \*A, const size\_t lda, const [FFLAS\\_ELT](#) \*B, const size\_t ldb, const [FFLAS\\_ELT](#) beta, [FFLAS\\_ELT](#) \*C, const size\_t ldc, const [ParSeqHelper::Sequential](#) seq)

- template `INST_OR_DECL FFLAS_ELT * fgemm` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS_TRANSPOSE` ta, const `FFLAS_TRANSPOSE` tb, const `size_t` m, const `size_t` n, const `size_t` k, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*A, const `size_t` lda, const `FFLAS_ELT` \*B, const `size_t` ldb, const `FFLAS_ELT` beta, `FFLAS_ELT` \*C, const `size_t` ldc, const `ParSeqHelper::Parallel< CuttingStrategy::Recursive, StrategyParameter::TwoDAdaptive >` par)
- template `INST_OR_DECL FFLAS_ELT * fgemm` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS_TRANSPOSE` ta, const `FFLAS_TRANSPOSE` tb, const `size_t` m, const `size_t` n, const `size_t` k, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*A, const `size_t` lda, const `FFLAS_ELT` \*B, const `size_t` ldb, const `FFLAS_ELT` beta, `FFLAS_ELT` \*C, const `size_t` ldc, const `ParSeqHelper::Parallel< CuttingStrategy::Block, StrategyParameter::Threads >` par)
- template `INST_OR_DECL FFLAS_ELT * fsquare` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS_TRANSPOSE` ta, const `size_t` n, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*A, const `size_t` lda, const `FFLAS_ELT` beta, `FFLAS_ELT` \*C, const `size_t` ldc)

*fsquare: Squares a matrix.*

## 13.228.1 Macro Definition Documentation

### 13.228.1.1 \_\_FFLAS\_\_TRSM\_READONLY

```
#define __FFLAS__TRSM_READONLY
```

## 13.229 fflas\_lvl1.C File Reference

C functions calls for level 1 `FFLAS` in `fflas-c.h`.

```
#include "fflas-ffpack/interfaces/libs/fflas_c.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "givaro//modular-balanced.h"
#include "givaro//modular.h"
```

### Functions

- void `freducein_1_modular_double` (const double p, const `size_t` n, double \*X, const `size_t` incX, bool positive)
- void `freduce_1_modular_double` (const double p, const `size_t` n, const double \*Y, const `size_t` incY, double \*X, const `size_t` incX, bool positive)
- void `fnegin_1_modular_double` (const double p, const `size_t` n, double \*X, const `size_t` incX, bool positive)
- void `fneg_1_modular_double` (const double p, const `size_t` n, const double \*Y, const `size_t` incY, double \*X, const `size_t` incX, bool positive)
- void `fzero_1_modular_double` (const double p, const `size_t` n, double \*X, const `size_t` incX, bool positive)
- bool `fiszero_1_modular_double` (const double p, const `size_t` n, const double \*X, const `size_t` incX, bool positive)
- bool `fequal_1_modular_double` (const double p, const `size_t` n, const double \*X, const `size_t` incX, const double \*Y, const `size_t` incY, bool positive)
- void `fassign_1_modular_double` (const double p, const `size_t` n, const double \*Y, const `size_t` incY, double \*X, const `size_t` incX, bool positive)
- void `fscal_1_modular_double` (const double p, const `size_t` n, const double alpha, double \*X, const `size_t` incX, bool positive)
- void `fscale_1_modular_double` (const double p, const `size_t` n, const double alpha, const double \*X, const `size_t` incX, double \*Y, const `size_t` incY, bool positive)
- void `faxpy_1_modular_double` (const double p, const `size_t` n, const double alpha, const double \*X, const `size_t` incX, double \*Y, const `size_t` incY, bool positive)
- double `fdot_1_modular_double` (const double p, const `size_t` n, const double \*X, const `size_t` incX, const double \*Y, const `size_t` incY, bool positive)
- void `fswap_1_modular_double` (const double p, const `size_t` n, double \*X, const `size_t` incX, double \*Y, const `size_t` incY, bool positive)
- void `fadd_1_modular_double` (const double p, const `size_t` n, const double \*A, const `size_t` incA, const double \*B, const `size_t` incB, double \*C, const `size_t` incC, bool positive)

- void [fsub\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*A, const size\_t incA, const double \*B, const size\_t incB, double \*C, const size\_t incC, bool positive)
- void [faddin\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*B, const size\_t incB, double \*C, const size\_t incC, bool positive)
- void [fsubin\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*B, const size\_t incB, double \*C, const size\_t incC, bool positive)

### 13.229.1 Detailed Description

C functions calls for level 1 [FFLAS](#) in [fflas-c.h](#).

Author

Brice Boyer

See also

[fflas/fflas\\_level1.inl](#)

### 13.229.2 Function Documentation

#### 13.229.2.1 [freducein\\_1\\_modular\\_double\(\)](#)

```
void freducein_1_modular_double (
    const double p,
    const size_t n,
    double * X,
    const size_t incX,
    bool positive)
```

#### 13.229.2.2 [freduce\\_1\\_modular\\_double\(\)](#)

```
void freduce_1_modular_double (
    const double p,
    const size_t n,
    const double * Y,
    const size_t incY,
    double * X,
    const size_t incX,
    bool positive)
```

#### 13.229.2.3 [fnegin\\_1\\_modular\\_double\(\)](#)

```
void fnegin_1_modular_double (
    const double p,
    const size_t n,
    double * X,
    const size_t incX,
    bool positive)
```

#### 13.229.2.4 [fneg\\_1\\_modular\\_double\(\)](#)

```
void fneg_1_modular_double (
    const double p,
    const size_t n,
    const double * Y,
    const size_t incY,
    double * X,
    const size_t incX,
    bool positive)
```

**13.229.2.5 fzero\_1\_modular\_double()**

```
void fzero_1_modular_double (
    const double p,
    const size_t n,
    double * X,
    const size_t incX,
    bool positive)
```

**13.229.2.6 fiszero\_1\_modular\_double()**

```
bool fiszero_1_modular_double (
    const double p,
    const size_t n,
    const double * X,
    const size_t incX,
    bool positive)
```

**13.229.2.7 fequal\_1\_modular\_double()**

```
bool fequal_1_modular_double (
    const double p,
    const size_t n,
    const double * X,
    const size_t incX,
    const double * Y,
    const size_t incY,
    bool positive)
```

**13.229.2.8 fassign\_1\_modular\_double()**

```
void fassign_1_modular_double (
    const double p,
    const size_t n,
    const double * Y,
    const size_t incY,
    double * X,
    const size_t incX,
    bool positive)
```

**13.229.2.9 fscal\_1\_modular\_double()**

```
void fscal_1_modular_double (
    const double p,
    const size_t n,
    const double alpha,
    double * X,
    const size_t incX,
    bool positive)
```

**13.229.2.10 fscale\_1\_modular\_double()**

```
void fscale_1_modular_double (
    const double p,
    const size_t n,
    const double alpha,
    const double * X,
    const size_t incX,
    double * Y,
```

```
    const size_t incY,  
    bool positive)
```

#### 13.229.2.11 faxpy\_1\_modular\_double()

```
void faxpy_1_modular_double (  
    const double p,  
    const size_t n,  
    const double alpha,  
    const double * X,  
    const size_t incX,  
    double * Y,  
    const size_t incY,  
    bool positive)
```

#### 13.229.2.12 fdot\_1\_modular\_double()

```
double fdot_1_modular_double (  
    const double p,  
    const size_t n,  
    const double * X,  
    const size_t incX,  
    const double * Y,  
    const size_t incY,  
    bool positive)
```

#### 13.229.2.13 fswap\_1\_modular\_double()

```
void fswap_1_modular_double (  
    const double p,  
    const size_t n,  
    double * X,  
    const size_t incX,  
    double * Y,  
    const size_t incY,  
    bool positive)
```

#### 13.229.2.14 fadd\_1\_modular\_double()

```
void fadd_1_modular_double (  
    const double p,  
    const size_t n,  
    const double * A,  
    const size_t incA,  
    const double * B,  
    const size_t incB,  
    double * C,  
    const size_t incC,  
    bool positive)
```

#### 13.229.2.15 fsub\_1\_modular\_double()

```
void fsub_1_modular_double (  
    const double p,  
    const size_t n,  
    const double * A,  
    const size_t incA,  
    const double * B,  
    const size_t incB,
```

```
double * C,
const size_t incC,
bool positive)
```

### 13.229.2.16 faddin\_1\_modular\_double()

```
void faddin_1_modular_double (
    const double p,
    const size_t n,
    const double * B,
    const size_t incB,
    double * C,
    const size_t incC,
    bool positive)
```

### 13.229.2.17 fsubin\_1\_modular\_double()

```
void fsubin_1_modular_double (
    const double p,
    const size_t n,
    const double * B,
    const size_t incB,
    double * C,
    const size_t incC,
    bool positive)
```

## 13.230 fflas\_lvl2.C File Reference

C functions calls for level 2 [FFLAS](#) in fflas-c.h.

```
#include "fflas-ffpack/interfaces/libs/fflas_c.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "givaro/modular-balanced.h"
#include "givaro/modular.h"
```

### Functions

- void [fassign\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*A, const size\_t lda, double \*B, const size\_t ldb, bool positive)
- void [fzero\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, double \*A, const size\_t lda, bool positive)
- bool [fequal\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*A, const size\_t lda, const double \*B, const size\_t ldb, bool positive)
- bool [fiszero\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*A, const size\_t lda, bool positive)
- void [fidentity\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, double \*A, const size\_t lda, const double d, bool positive)
- void [freducein\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, double \*A, const size\_t lda, bool positive)
- void [freduce\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*A, const size\_t lda, double \*B, const size\_t ldb, bool positive)
- void [fnegin\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, double \*A, const size\_t lda, bool positive)
- void [fneg\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*A, const size\_t lda, double \*B, const size\_t ldb, bool positive)
- void [fscalin\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double alpha, double \*A, const size\_t lda, bool positive)

- void [fscal\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double alpha, const double \*A, const size\_t lda, double \*B, const size\_t ldb, bool positive)
- void [faxpy\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double alpha, const double \*A, const size\_t lda, double \*B, const size\_t ldb, bool positive)
- void [fmove\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, double \*A, const size\_t lda, double \*B, const size\_t ldb, bool positive)
- void [fadd\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*A, const size\_t lda, const double \*B, const size\_t ldb, double \*C, const size\_t ldc, bool positive)
- void [fsub\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*A, const size\_t lda, const double \*B, const size\_t ldb, double \*C, const size\_t ldc, bool positive)
- void [fsubin\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*B, const size\_t ldb, double \*C, const size\_t ldc, bool positive)
- void [faddin\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*B, const size\_t ldb, double \*C, const size\_t ldc, bool positive)
- double \* [fgemv\\_2\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_TRANSPOSE](#) TransA, const size\_t m, const size\_t n, const double alpha, const double \*A, const size\_t lda, const double \*X, const size\_t incX, const double beta, double \*Y, const size\_t incY, bool positive)
- void [fger\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double alpha, const double \*X, const size\_t incX, const double \*Y, const size\_t incY, double \*A, const size\_t lda, bool positive)
- void [ftrsv\\_2\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const enum [FFLAS\\_C\\_TRANSPOSE](#) TransA, const enum [FFLAS\\_C\\_DIAG](#) Diag, const size\_t n, const double \*A, const size\_t lda, double \*X, int incX, bool positive)

### 13.230.1 Detailed Description

C functions calls for level 2 [FFLAS](#) in `fflas-c.h`.

Author

Brice Boyer

See also

[fflas/fflas\\_level2.inl](#)

### 13.230.2 Function Documentation

#### 13.230.2.1 `fassign_2_modular_double()`

```
void fassign_2_modular_double (
    const double p,
    const size_t m,
    const size_t n,
    const double * A,
    const size_t lda,
    double * B,
    const size_t ldb,
    bool positive)
```

#### 13.230.2.2 `fzero_2_modular_double()`

```
void fzero_2_modular_double (
    const double p,
    const size_t m,
    const size_t n,
    double * A,
    const size_t lda,
    bool positive)
```



### 13.230.2.3 fequal\_2\_modular\_double()

```
bool fequal_2_modular_double (
    const double p,
    const size_t m,
    const size_t n,
    const double * A,
    const size_t lda,
    const double * B,
    const size_t ldb,
    bool positive)
```

### 13.230.2.4 fiszero\_2\_modular\_double()

```
bool fiszero_2_modular_double (
    const double p,
    const size_t m,
    const size_t n,
    const double * A,
    const size_t lda,
    bool positive)
```

### 13.230.2.5 fidentity\_2\_modular\_double()

```
void fidentity_2_modular_double (
    const double p,
    const size_t m,
    const size_t n,
    double * A,
    const size_t lda,
    const double d,
    bool positive)
```

### 13.230.2.6 freducein\_2\_modular\_double()

```
void freducein_2_modular_double (
    const double p,
    const size_t m,
    const size_t n,
    double * A,
    const size_t lda,
    bool positive)
```

### 13.230.2.7 freduce\_2\_modular\_double()

```
void freduce_2_modular_double (
    const double p,
    const size_t m,
    const size_t n,
    const double * A,
    const size_t lda,
    double * B,
    const size_t ldb,
    bool positive)
```

### 13.230.2.8 fnegin\_2\_modular\_double()

```
void fnegin_2_modular_double (
    const double p,
```

```
    const size_t m,  
    const size_t n,  
    double * A,  
    const size_t lda,  
    bool positive)
```

#### 13.230.2.9 fneg\_2\_modular\_double()

```
void fneg_2_modular_double (  
    const double p,  
    const size_t m,  
    const size_t n,  
    const double * A,  
    const size_t lda,  
    double * B,  
    const size_t ldb,  
    bool positive)
```

#### 13.230.2.10 fscaln\_2\_modular\_double()

```
void fscaln_2_modular_double (  
    const double p,  
    const size_t m,  
    const size_t n,  
    const double alpha,  
    double * A,  
    const size_t lda,  
    bool positive)
```

#### 13.230.2.11 fscal\_2\_modular\_double()

```
void fscal_2_modular_double (  
    const double p,  
    const size_t m,  
    const size_t n,  
    const double alpha,  
    const double * A,  
    const size_t lda,  
    double * B,  
    const size_t ldb,  
    bool positive)
```

#### 13.230.2.12 faxpy\_2\_modular\_double()

```
void faxpy_2_modular_double (  
    const double p,  
    const size_t m,  
    const size_t n,  
    const double alpha,  
    const double * A,  
    const size_t lda,  
    double * B,  
    const size_t ldb,  
    bool positive)
```

#### 13.230.2.13 fmove\_2\_modular\_double()

```
void fmove_2_modular_double (  
    const double p,
```

```
    const size_t m,  
    const size_t n,  
    double * A,  
    const size_t lda,  
    double * B,  
    const size_t ldb,  
    bool positive)
```

#### 13.230.2.14 fadd\_2\_modular\_double()

```
void fadd_2_modular_double (  
    const double p,  
    const size_t m,  
    const size_t n,  
    const double * A,  
    const size_t lda,  
    const double * B,  
    const size_t ldb,  
    double * C,  
    const size_t ldc,  
    bool positive)
```

#### 13.230.2.15 fsub\_2\_modular\_double()

```
void fsub_2_modular_double (  
    const double p,  
    const size_t m,  
    const size_t n,  
    const double * A,  
    const size_t lda,  
    const double * B,  
    const size_t ldb,  
    double * C,  
    const size_t ldc,  
    bool positive)
```

#### 13.230.2.16 fsubin\_2\_modular\_double()

```
void fsubin_2_modular_double (  
    const double p,  
    const size_t m,  
    const size_t n,  
    const double * B,  
    const size_t ldb,  
    double * C,  
    const size_t ldc,  
    bool positive)
```

#### 13.230.2.17 faddin\_2\_modular\_double()

```
void faddin_2_modular_double (  
    const double p,  
    const size_t m,  
    const size_t n,  
    const double * B,  
    const size_t ldb,  
    double * C,  
    const size_t ldc,  
    bool positive)
```

**13.230.2.18 fgemv\_2\_modular\_double()**

```
double * fgemv_2_modular_double (
    const double p,
    const enum FFLAS_C_TRANSPOSE TransA,
    const size_t m,
    const size_t n,
    const double alpha,
    const double * A,
    const size_t lda,
    const double * X,
    const size_t incX,
    const double beta,
    double * Y,
    const size_t incY,
    bool positive)
```

**13.230.2.19 fger\_2\_modular\_double()**

```
void fger_2_modular_double (
    const double p,
    const size_t m,
    const size_t n,
    const double alpha,
    const double * X,
    const size_t incX,
    const double * Y,
    const size_t incY,
    double * A,
    const size_t lda,
    bool positive)
```

**13.230.2.20 ftrsv\_2\_modular\_double()**

```
void ftrsv_2_modular_double (
    const double p,
    const enum FFLAS_C_UPLO Uplo,
    const enum FFLAS_C_TRANSPOSE TransA,
    const enum FFLAS_C_DIAG Diag,
    const size_t n,
    const double * A,
    const size_t lda,
    double * X,
    int incX,
    bool positive)
```

**13.231 fflas\_lvl3.C File Reference**

C functions calls for level 3 [FFLAS](#) in `fflas-c.h`.

```
#include "fflas-ffpack/interfaces/libs/fflas_c.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "givaro//modular-balanced.h"
#include "givaro//modular.h"
```

**Functions**

- void [ftrsm\\_3\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_SIDE](#) Side, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const enum [FFLAS\\_C\\_TRANSPOSE](#) tA, const enum [FFLAS\\_C\\_DIAG](#) Diag, const

- size\_t m, const size\_t n, const double alpha, const double \*A, const size\_t ldA, double \*B, const size\_t ldB, bool positive)
- void [ftrmm\\_3\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_SIDE](#) Side, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const enum [FFLAS\\_C\\_TRANSPOSE](#) tA, const enum [FFLAS\\_C\\_DIAG](#) Diag, const size\_t m, const size\_t n, const double alpha, double \*A, const size\_t ldA, double \*B, const size\_t ldB, bool positive)
  - double \* [fgemm\\_3\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_TRANSPOSE](#) tA, const enum [FFLAS\\_C\\_TRANSPOSE](#) tB, const size\_t m, const size\_t n, const size\_t k, const double alpha, const double \*A, const size\_t ldA, const double \*B, const size\_t ldB, const double betA, double \*C, const size\_t ldC, bool positive)
  - double \* [fsquare\\_3\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_TRANSPOSE](#) tA, const size\_t n, const double alpha, const double \*A, const size\_t ldA, const double betA, double \*C, const size\_t ldC, bool positive)

### 13.231.1 Detailed Description

C functions calls for level 3 [FFLAS](#) in fflas-c.h.

Author

Brice Boyer

See also

[fflas/fflas\\_level3.inl](#)

### 13.231.2 Function Documentation

#### 13.231.2.1 ftrsm\_3\_modular\_double()

```
void ftrsm_3_modular_double (
    const double p,
    const enum FFLAS\_C\_SIDE Side,
    const enum FFLAS\_C\_UPLO Uplo,
    const enum FFLAS\_C\_TRANSPOSE tA,
    const enum FFLAS\_C\_DIAG Diag,
    const size_t m,
    const size_t n,
    const double alpha,
    const double * A,
    const size_t ldA,
    double * B,
    const size_t ldB,
    bool positive)
```

#### 13.231.2.2 ftrmm\_3\_modular\_double()

```
void ftrmm_3_modular_double (
    const double p,
    const enum FFLAS\_C\_SIDE Side,
    const enum FFLAS\_C\_UPLO Uplo,
    const enum FFLAS\_C\_TRANSPOSE tA,
    const enum FFLAS\_C\_DIAG Diag,
    const size_t m,
    const size_t n,
    const double alpha,
    double * A,
    const size_t ldA,
    double * B,
    const size_t ldB,
    bool positive)
```

### 13.231.2.3 fgemm\_3\_modular\_double()

```
double * fgemm_3_modular_double (
    const double p,
    const enum FFLAS_C_TRANSPOSE tA,
    const enum FFLAS_C_TRANSPOSE tB,
    const size_t m,
    const size_t n,
    const size_t k,
    const double alpha,
    const double * A,
    const size_t ldA,
    const double * B,
    const size_t ldB,
    const double betaA,
    double * C,
    const size_t ldC,
    bool positive)
```

### 13.231.2.4 fsquare\_3\_modular\_double()

```
double * fsquare_3_modular_double (
    const double p,
    const enum FFLAS_C_TRANSPOSE tA,
    const size_t n,
    const double alpha,
    const double * A,
    const size_t ldA,
    const double betaA,
    double * C,
    const size_t ldC,
    bool positive)
```

## 13.232 fflas\_sparse.C File Reference

C functions calls for level 1.5 and 2.5 [FFLAS](#) in fflas-c.h.

### 13.232.1 Detailed Description

C functions calls for level 1.5 and 2.5 [FFLAS](#) in fflas-c.h.

#### Author

Brice Boyer

#### See also

[fflas/fflas\\_sparse.h](#)

## 13.233 ffpack.C File Reference

C functions calls for [FFPACK](#) in ffpack-c.h.

```
#include "fflas-ffpack/interfaces/libs/ffpack_c.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "givaro/modular-balanced.h"
#include "givaro/modular.h"
```

## Functions

- void [LAPACKPerm2MathPerm](#) (size\_t \*MathP, const size\_t \*LapackP, const size\_t N)
- void [MathPerm2LAPACKPerm](#) (size\_t \*LapackP, const size\_t \*MathP, const size\_t N)
- void [MatrixApplyS\\_modular\\_double](#) (const double p, double \*A, const size\_t lda, const size\_t width, const size\_t M2, const size\_t R1, const size\_t R2, const size\_t R3, const size\_t R4, bool positive)
- void [PermApplyS\\_double](#) (double \*A, const size\_t lda, const size\_t width, const size\_t M2, const size\_t R1, const size\_t R2, const size\_t R3, const size\_t R4)
- void [MatrixApplyT\\_modular\\_double](#) (const double p, double \*A, const size\_t lda, const size\_t width, const size\_t N2, const size\_t R1, const size\_t R2, const size\_t R3, const size\_t R4, bool positive)
- void [PermApplyT\\_double](#) (double \*A, const size\_t lda, const size\_t width, const size\_t N2, const size\_t R1, const size\_t R2, const size\_t R3, const size\_t R4)
- void [composePermutationsLLM](#) (size\_t \*MathP, const size\_t \*P1, const size\_t \*P2, const size\_t R, const size\_t N)
- void [composePermutationsLLL](#) (size\_t \*P1, const size\_t \*P2, const size\_t R, const size\_t N)
- void [composePermutationsMLM](#) (size\_t \*MathP1, const size\_t \*P2, const size\_t R, const size\_t N)
- void [cyclic\\_shift\\_mathPerm](#) (size\_t \*P, const size\_t s)
- void [cyclic\\_shift\\_row\\_modular\\_double](#) (const double p, double \*A, size\_t m, size\_t n, size\_t lda, bool positive)
- void [cyclic\\_shift\\_col\\_modular\\_double](#) (const double p, double \*A, size\_t m, size\_t n, size\_t lda, bool positive)
- void [applyP\\_modular\\_double](#) (const double p, const enum [FFLAS::FFLAS\\_SIDE](#) Side, const enum [FFLAS::FFLAS\\_TRANSPOSE](#) Trans, const size\_t M, const size\_t ibeg, const size\_t iend, double \*A, const size\_t lda, const size\_t \*P, bool positive)
- void [fgetrsin\\_modular\\_double](#) (const double p, const enum [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t R, double \*A, const size\_t lda, const size\_t \*P, const size\_t \*Q, double \*B, const size\_t ldb, int \*info, bool positive)
- double \* [fgetrsv\\_modular\\_double](#) (const double p, const enum [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t NRHS, const size\_t R, double \*A, const size\_t lda, const size\_t \*P, const size\_t \*Q, double \*X, const size\_t ldx, const double \*B, const size\_t ldb, int \*info, bool positive)
- size\_t [fgesvin\\_modular\\_double](#) (const double p, const enum [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, double \*A, const size\_t lda, double \*B, const size\_t ldb, int \*info, bool positive)
- size\_t [fgesv\\_modular\\_double](#) (const double p, const enum [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t NRHS, double \*A, const size\_t lda, double \*X, const size\_t ldx, const double \*B, const size\_t ldb, int \*info, bool positive)
- void [ftrtri\\_modular\\_double](#) (const double p, const enum [FFLAS::FFLAS\\_UPLO](#) Uplo, const enum [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t N, double \*A, const size\_t lda, bool positive)
- void [trinv\\_left\\_modular\\_double](#) (const double p, const size\_t N, const double \*L, const size\_t ldl, double \*X, const size\_t ldx, bool positive)
- void [ftrtm\\_modular\\_double](#) (const double p, const [FFLAS::FFLAS\\_SIDE](#) side, const enum [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t N, double \*A, const size\_t lda, bool positive)
- size\_t [PLUQ\\_modular\\_double](#) (const double p, const enum [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Q, bool positive)
- size\_t [LUdivine\\_modular\\_double](#) (const double p, const enum [FFLAS::FFLAS\\_DIAG](#) Diag, const enum [FFLAS::FFLAS\\_TRANSPOSE](#) Trans, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, const size\_t cutoff, bool positive)
- size\_t [ColumnEchelonForm\\_modular\\_double](#) (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, bool transform, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- size\_t [RowEchelonForm\\_modular\\_double](#) (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- size\_t [ReducedColumnEchelonForm\\_modular\\_double](#) (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- size\_t [ReducedRowEchelonForm\\_modular\\_double](#) (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- size\_t [ColumnEchelonForm\\_modular\\_float](#) (const float p, const size\_t M, const size\_t N, float \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, bool transform, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)

- `size_t RowEchelonForm_modular_float` (const float p, const size\_t M, const size\_t N, float \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t ReducedColumnEchelonForm_modular_float` (const float p, const size\_t M, const size\_t N, float \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t ReducedRowEchelonForm_modular_float` (const float p, const size\_t M, const size\_t N, float \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t ColumnEchelonForm_modular_int32_t` (const int32\_t p, const size\_t M, const size\_t N, int32\_t \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t RowEchelonForm_modular_int32_t` (const int32\_t p, const size\_t M, const size\_t N, int32\_t \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t ReducedColumnEchelonForm_modular_int32_t` (const int32\_t p, const size\_t M, const size\_t N, int32\_t \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t ReducedRowEchelonForm_modular_int32_t` (const int32\_t p, const size\_t M, const size\_t N, int32\_t \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t pColumnEchelonForm_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t pRowEchelonForm_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t pReducedColumnEchelonForm_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t pReducedRowEchelonForm_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t pColumnEchelonForm_modular_float` (const float p, const size\_t M, const size\_t N, float \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t pRowEchelonForm_modular_float` (const float p, const size\_t M, const size\_t N, float \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t pReducedColumnEchelonForm_modular_float` (const float p, const size\_t M, const size\_t N, float \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t pReducedRowEchelonForm_modular_float` (const float p, const size\_t M, const size\_t N, float \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t pColumnEchelonForm_modular_int32_t` (const int32\_t p, const size\_t M, const size\_t N, int32\_t \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t pRowEchelonForm_modular_int32_t` (const int32\_t p, const size\_t M, const size\_t N, int32\_t \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t pReducedColumnEchelonForm_modular_int32_t` (const int32\_t p, const size\_t M, const size\_t N, int32\_t \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t pReducedRowEchelonForm_modular_int32_t` (const int32\_t p, const size\_t M, const size\_t N, int32\_t \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `double * Invertin_modular_double` (const double p, const size\_t M, double \*A, const size\_t lda, int \*nullity, bool positive)
- `double * Invert_modular_double` (const double p, const size\_t M, const double \*A, const size\_t lda, double \*X, const size\_t ldx, int \*nullity, bool positive)
- `double * Invert2_modular_double` (const double p, const size\_t M, double \*A, const size\_t lda, double \*X, const size\_t ldx, int \*nullity, bool positive)
- `size_t KrylovElim_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Q, const size\_t deg, size\_t \*iterates, size\_t \*inviterates, const size\_t maxit, size\_t virt, bool positive)
- `size_t SpecRankProfile_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, const size\_t deg, size\_t \*rankProfile, bool positive)



- `size_t Rank_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, bool positive)
- `bool IsSingular_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, bool positive)
- `double Det_modular_double` (const double p, const size\_t N, double \*A, const size\_t lda, bool positive)
- `double * Solve_modular_double` (const double p, const size\_t M, double \*A, const size\_t lda, double \*x, const int incx, const double \*b, const int incb, bool positive)
- `void solveLB_modular_double` (const double p, const enum FFLAS::FFLAS\_SIDE Side, const size\_t M, const size\_t N, const size\_t R, double \*L, const size\_t ldl, const size\_t \*Q, double \*B, const size\_t ldb, bool positive)
- `void solveLB2_modular_double` (const double p, const enum FFLAS::FFLAS\_SIDE Side, const size\_t M, const size\_t N, const size\_t R, double \*L, const size\_t ldl, const size\_t \*Q, double \*B, const size\_t ldb, bool positive)
- `void RandomNullSpaceVector_modular_double` (const double p, const enum FFLAS::FFLAS\_SIDE Side, const size\_t M, const size\_t N, double \*A, const size\_t lda, double \*X, const size\_t incX, bool positive)
- `size_t NullSpaceBasis_modular_double` (const double p, const enum FFLAS::FFLAS\_SIDE Side, const size\_t M, const size\_t N, double \*A, const size\_t lda, double \*\*NS, size\_t \*ldn, size\_t \*NSdim, bool positive)
- `size_t RowRankProfile_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*\*rkprofile, const enum FFPACK\_C\_LU\_TAG LuTag, bool positive)
- `size_t ColumnRankProfile_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*\*rkprofile, const enum FFPACK\_C\_LU\_TAG LuTag, bool positive)
- `void RankProfileFromLU` (const size\_t \*P, const size\_t N, const size\_t R, size\_t \*\*rkprofile, const enum FFPACK\_C\_LU\_TAG LuTag)
- `size_t LeadingSubmatrixRankProfiles` (const size\_t M, const size\_t N, const size\_t R, const size\_t LSm, const size\_t LSn, const size\_t \*P, const size\_t \*Q, size\_t \*RRP, size\_t \*CRP)
- `size_t RowRankProfileSubmatrixIndices_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*\*rowindices, size\_t \*\*colindices, size\_t \*R, bool positive)
- `size_t ColRankProfileSubmatrixIndices_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*\*rowindices, size\_t \*\*colindices, size\_t \*R, bool positive)
- `size_t RowRankProfileSubmatrix_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, double \*\*X, size\_t \*R, bool positive)
- `size_t ColRankProfileSubmatrix_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, double \*\*X, size\_t \*R, bool positive)
- `void getTriangular_modular_double` (const double p, const enum FFLAS::FFLAS\_UPLO Uplo, const enum FFLAS::FFLAS\_DIAG Diag, const size\_t M, const size\_t N, const size\_t R, const double \*A, const size\_t lda, double \*T, const size\_t ldt, const bool OnlyNonZeroVectors, bool positive)
- `void getTriangularin_modular_double` (const double p, const enum FFLAS::FFLAS\_UPLO Uplo, const enum FFLAS::FFLAS\_DIAG Diag, const size\_t M, const size\_t N, const size\_t R, double \*A, const size\_t lda, bool positive)
- `void getEchelonForm_modular_double` (const double p, const enum FFLAS::FFLAS\_UPLO Uplo, const enum FFLAS::FFLAS\_DIAG Diag, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const double \*A, const size\_t lda, double \*T, const size\_t ldt, const bool OnlyNonZeroVectors, const enum FFPACK\_C\_LU\_TAG LuTag, bool positive)
- `void getEchelonFormin_modular_double` (const double p, const enum FFLAS::FFLAS\_UPLO Uplo, const enum FFLAS::FFLAS\_DIAG Diag, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, double \*A, const size\_t lda, const enum FFPACK\_C\_LU\_TAG LuTag, bool positive)
- `void getEchelonTransform_modular_double` (const double p, const enum FFLAS::FFLAS\_UPLO Uplo, const enum FFLAS::FFLAS\_DIAG Diag, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const size\_t \*Q, const double \*A, const size\_t lda, double \*T, const size\_t ldt, const enum FFPACK\_C\_LU\_TAG LuTag, bool positive)
- `void getReducedEchelonForm_modular_double` (const double p, const enum FFLAS::FFLAS\_UPLO Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const double \*A, const size\_t lda, double \*T, const size\_t ldt, const bool OnlyNonZeroVectors, const enum FFPACK\_C\_LU\_TAG LuTag, bool positive)
- `void getReducedEchelonFormin_modular_double` (const double p, const enum FFLAS::FFLAS\_UPLO Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, double \*A, const size\_t lda, const enum FFPACK\_C\_LU\_TAG LuTag, bool positive)

- void [getReducedEchelonTransform\\_modular\\_double](#) (const double p, const enum [FFLAS::FFLAS\\_UPLO](#) Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const size\_t \*Q, const double \*A, const size\_t lda, double \*T, const size\_t ldt, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- void [PLUQtoEchelonPermutation](#) (const size\_t N, const size\_t R, const size\_t \*P, size\_t \*outPerm)

### 13.233.1 Detailed Description

C functions calls for [FFPACK](#) in `ffpack-c.h`.

Author

Brice Boyer

See also

[ffpack/ffpack.h](#)

### 13.233.2 Function Documentation

#### 13.233.2.1 LAPACKPerm2MathPerm()

```
void LAPACKPerm2MathPerm (
    size_t * MathP,
    const size_t * LapackP,
    const size_t N)
```

#### 13.233.2.2 MathPerm2LAPACKPerm()

```
void MathPerm2LAPACKPerm (
    size_t * LapackP,
    const size_t * MathP,
    const size_t N)
```

#### 13.233.2.3 MatrixApplyS\_modular\_double()

```
void MatrixApplyS_modular_double (
    const double p,
    double * A,
    const size_t lda,
    const size_t width,
    const size_t M2,
    const size_t R1,
    const size_t R2,
    const size_t R3,
    const size_t R4,
    bool positive)
```

#### 13.233.2.4 PermApplyS\_double()

```
void PermApplyS_double (
    double * A,
    const size_t lda,
    const size_t width,
    const size_t M2,
    const size_t R1,
    const size_t R2,
    const size_t R3,
    const size_t R4)
```

### 13.233.2.5 MatrixApplyT\_modular\_double()

```
void MatrixApplyT_modular_double (
    const double p,
    double * A,
    const size_t lda,
    const size_t width,
    const size_t N2,
    const size_t R1,
    const size_t R2,
    const size_t R3,
    const size_t R4,
    bool positive)
```

### 13.233.2.6 PermApplyT\_double()

```
void PermApplyT_double (
    double * A,
    const size_t lda,
    const size_t width,
    const size_t N2,
    const size_t R1,
    const size_t R2,
    const size_t R3,
    const size_t R4)
```

### 13.233.2.7 composePermutationsLLM()

```
void composePermutationsLLM (
    size_t * MathP,
    const size_t * P1,
    const size_t * P2,
    const size_t R,
    const size_t N)
```

### 13.233.2.8 composePermutationsLLL()

```
void composePermutationsLLL (
    size_t * P1,
    const size_t * P2,
    const size_t R,
    const size_t N)
```

### 13.233.2.9 composePermutationsMLM()

```
void composePermutationsMLM (
    size_t * MathP1,
    const size_t * P2,
    const size_t R,
    const size_t N)
```

### 13.233.2.10 cyclic\_shift\_mathPerm()

```
void cyclic_shift_mathPerm (
    size_t * P,
    const size_t s)
```

**13.233.2.11 cyclic\_shift\_row\_modular\_double()**

```
void cyclic_shift_row_modular_double (
    const double p,
    double * A,
    size_t m,
    size_t n,
    size_t lda,
    bool positive)
```

**13.233.2.12 cyclic\_shift\_col\_modular\_double()**

```
void cyclic_shift_col_modular_double (
    const double p,
    double * A,
    size_t m,
    size_t n,
    size_t lda,
    bool positive)
```

**13.233.2.13 applyP\_modular\_double()**

```
void applyP_modular_double (
    const double p,
    const enum FFLAS::FFLAS_SIDE Side,
    const enum FFLAS::FFLAS_TRANSPOSE Trans,
    const size_t M,
    const size_t ibeg,
    const size_t iend,
    double * A,
    const size_t lda,
    const size_t * P,
    bool positive)
```

**13.233.2.14 fgetrsin\_modular\_double()**

```
void fgetrsin_modular_double (
    const double p,
    const enum FFLAS::FFLAS_SIDE Side,
    const size_t M,
    const size_t N,
    const size_t R,
    double * A,
    const size_t lda,
    const size_t * P,
    const size_t * Q,
    double * B,
    const size_t ldb,
    int * info,
    bool positive)
```

**13.233.2.15 fgetrsv\_modular\_double()**

```
double * fgetrsv_modular_double (
    const double p,
    const enum FFLAS::FFLAS_SIDE Side,
    const size_t M,
    const size_t N,
    const size_t NRHS,
```

```
const size_t R,  
double * A,  
const size_t lda,  
const size_t * P,  
const size_t * Q,  
double * X,  
const size_t ldX,  
const double * B,  
const size_t ldb,  
int * info,  
bool positive)
```

#### 13.233.2.16 fgesvin\_modular\_double()

```
size_t fgesvin_modular_double (  
    const double p,  
    const enum FFLAS::FFLAS_SIDE Side,  
    const size_t M,  
    const size_t N,  
    double * A,  
    const size_t lda,  
    double * B,  
    const size_t ldb,  
    int * info,  
    bool positive)
```

#### 13.233.2.17 fgesv\_modular\_double()

```
size_t fgesv_modular_double (  
    const double p,  
    const enum FFLAS::FFLAS_SIDE Side,  
    const size_t M,  
    const size_t N,  
    const size_t NRHS,  
    double * A,  
    const size_t lda,  
    double * X,  
    const size_t ldX,  
    const double * B,  
    const size_t ldb,  
    int * info,  
    bool positive)
```

#### 13.233.2.18 ftrtri\_modular\_double()

```
void ftrtri_modular_double (  
    const double p,  
    const enum FFLAS::FFLAS_UPLO Uplo,  
    const enum FFLAS::FFLAS_DIAG Diag,  
    const size_t N,  
    double * A,  
    const size_t lda,  
    bool positive)
```

#### 13.233.2.19 trinv\_left\_modular\_double()

```
void trinv_left_modular_double (  
    const double p,  
    const size_t N,
```

```

    const double * L,
    const size_t ldl,
    double * X,
    const size_t ldx,
    bool positive)

```

#### 13.233.2.20 ftrtrm\_modular\_double()

```

void ftrtrm_modular_double (
    const double p,
    const FFLAS::FFLAS_SIDE side,
    const enum FFLAS::FFLAS_DIAG Diag,
    const size_t N,
    double * A,
    const size_t lda,
    bool positive)

```

#### 13.233.2.21 PLUQ\_modular\_double()

```

size_t PLUQ_modular_double (
    const double p,
    const enum FFLAS::FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t * P,
    size_t * Q,
    bool positive)

```

#### 13.233.2.22 LUdivine\_modular\_double()

```

size_t LUdivine_modular_double (
    const double p,
    const enum FFLAS::FFLAS_DIAG Diag,
    const enum FFLAS::FFLAS_TRANSPOSE Trans,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const enum FFPACK_C_LU_TAG LuTag,
    const size_t cutoff,
    bool positive)

```

#### 13.233.2.23 ColumnEchelonForm\_modular\_double()

```

size_t ColumnEchelonForm_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

**13.233.2.24 RowEchelonForm\_modular\_double()**

```
size_t RowEchelonForm_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)
```

**13.233.2.25 ReducedColumnEchelonForm\_modular\_double()**

```
size_t ReducedColumnEchelonForm_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)
```

**13.233.2.26 ReducedRowEchelonForm\_modular\_double()**

```
size_t ReducedRowEchelonForm_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)
```

**13.233.2.27 ColumnEchelonForm\_modular\_float()**

```
size_t ColumnEchelonForm_modular_float (
    const float p,
    const size_t M,
    const size_t N,
    float * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)
```

**13.233.2.28 RowEchelonForm\_modular\_float()**

```
size_t RowEchelonForm_modular_float (
    const float p,
```

```

    const size_t M,
    const size_t N,
    float * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

### 13.233.2.29 ReducedColumnEchelonForm\_modular\_float()

```

size_t ReducedColumnEchelonForm_modular_float (
    const float p,
    const size_t M,
    const size_t N,
    float * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

### 13.233.2.30 ReducedRowEchelonForm\_modular\_float()

```

size_t ReducedRowEchelonForm_modular_float (
    const float p,
    const size_t M,
    const size_t N,
    float * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

### 13.233.2.31 ColumnEchelonForm\_modular\_int32\_t()

```

size_t ColumnEchelonForm_modular_int32_t (
    const int32_t p,
    const size_t M,
    const size_t N,
    int32_t * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

### 13.233.2.32 RowEchelonForm\_modular\_int32\_t()

```

size_t RowEchelonForm_modular_int32_t (
    const int32_t p,
    const size_t M,
    const size_t N,
    int32_t * A,

```



```

    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

#### 13.233.2.33 ReducedColumnEchelonForm\_modular\_int32\_t()

```

size_t ReducedColumnEchelonForm_modular_int32_t (
    const int32_t p,
    const size_t M,
    const size_t N,
    int32_t * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

#### 13.233.2.34 ReducedRowEchelonForm\_modular\_int32\_t()

```

size_t ReducedRowEchelonForm_modular_int32_t (
    const int32_t p,
    const size_t M,
    const size_t N,
    int32_t * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

#### 13.233.2.35 pColumnEchelonForm\_modular\_double()

```

size_t pColumnEchelonForm_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

#### 13.233.2.36 pRowEchelonForm\_modular\_double()

```

size_t pRowEchelonForm_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,

```

```

    const bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

### 13.233.2.37 pReducedColumnEchelonForm\_modular\_double()

```

size_t pReducedColumnEchelonForm_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

### 13.233.2.38 pReducedRowEchelonForm\_modular\_double()

```

size_t pReducedRowEchelonForm_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

### 13.233.2.39 pColumnEchelonForm\_modular\_float()

```

size_t pColumnEchelonForm_modular_float (
    const float p,
    const size_t M,
    const size_t N,
    float * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

### 13.233.2.40 pRowEchelonForm\_modular\_float()

```

size_t pRowEchelonForm_modular_float (
    const float p,
    const size_t M,
    const size_t N,
    float * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

**13.233.2.41 pReducedColumnEchelonForm\_modular\_float()**

```
size_t pReducedColumnEchelonForm_modular_float (  
    const float p,  
    const size_t M,  
    const size_t N,  
    float * A,  
    const size_t lda,  
    size_t * P,  
    size_t * Qt,  
    const bool transform,  
    const enum FFPACK_C_LU_TAG LuTag,  
    bool positive)
```

**13.233.2.42 pReducedRowEchelonForm\_modular\_float()**

```
size_t pReducedRowEchelonForm_modular_float (  
    const float p,  
    const size_t M,  
    const size_t N,  
    float * A,  
    const size_t lda,  
    size_t * P,  
    size_t * Qt,  
    const bool transform,  
    const enum FFPACK_C_LU_TAG LuTag,  
    bool positive)
```

**13.233.2.43 pColumnEchelonForm\_modular\_int32\_t()**

```
size_t pColumnEchelonForm_modular_int32_t (  
    const int32_t p,  
    const size_t M,  
    const size_t N,  
    int32_t * A,  
    const size_t lda,  
    size_t * P,  
    size_t * Qt,  
    bool transform,  
    const enum FFPACK_C_LU_TAG LuTag,  
    bool positive)
```

**13.233.2.44 pRowEchelonForm\_modular\_int32\_t()**

```
size_t pRowEchelonForm_modular_int32_t (  
    const int32_t p,  
    const size_t M,  
    const size_t N,  
    int32_t * A,  
    const size_t lda,  
    size_t * P,  
    size_t * Qt,  
    const bool transform,  
    const enum FFPACK_C_LU_TAG LuTag,  
    bool positive)
```

**13.233.2.45 pReducedColumnEchelonForm\_modular\_int32\_t()**

```
size_t pReducedColumnEchelonForm_modular_int32_t (  
    const int32_t p,
```

```

    const size_t M,
    const size_t N,
    int32_t * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

#### 13.233.2.46 pReducedRowEchelonForm\_modular\_int32\_t()

```

size_t pReducedRowEchelonForm_modular_int32_t (
    const int32_t p,
    const size_t M,
    const size_t N,
    int32_t * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

#### 13.233.2.47 Invertin\_modular\_double()

```

double * Invertin_modular_double (
    const double p,
    const size_t M,
    double * A,
    const size_t lda,
    int * nullity,
    bool positive)

```

#### 13.233.2.48 Invert\_modular\_double()

```

double * Invert_modular_double (
    const double p,
    const size_t M,
    const double * A,
    const size_t lda,
    double * X,
    const size_t ldx,
    int * nullity,
    bool positive)

```

#### 13.233.2.49 Invert2\_modular\_double()

```

double * Invert2_modular_double (
    const double p,
    const size_t M,
    double * A,
    const size_t lda,
    double * X,
    const size_t ldx,
    int * nullity,
    bool positive)

```

**13.233.2.50 KrylovElim\_modular\_double()**

```
size_t KrylovElim_modular_double (  
    const double p,  
    const size_t M,  
    const size_t N,  
    double * A,  
    const size_t lda,  
    size_t * P,  
    size_t * Q,  
    const size_t deg,  
    size_t * iterates,  
    size_t * inviterates,  
    const size_t maxit,  
    size_t virt,  
    bool positive)
```

**13.233.2.51 SpecRankProfile\_modular\_double()**

```
size_t SpecRankProfile_modular_double (  
    const double p,  
    const size_t M,  
    const size_t N,  
    double * A,  
    const size_t lda,  
    const size_t deg,  
    size_t * rankProfile,  
    bool positive)
```

**13.233.2.52 Rank\_modular\_double()**

```
size_t Rank_modular_double (  
    const double p,  
    const size_t M,  
    const size_t N,  
    double * A,  
    const size_t lda,  
    bool positive)
```

**13.233.2.53 IsSingular\_modular\_double()**

```
bool IsSingular_modular_double (  
    const double p,  
    const size_t M,  
    const size_t N,  
    double * A,  
    const size_t lda,  
    bool positive)
```

**13.233.2.54 Det\_modular\_double()**

```
double Det_modular_double (  
    const double p,  
    const size_t N,  
    double * A,  
    const size_t lda,  
    bool positive)
```

**13.233.2.55 Solve\_modular\_double()**

```
double * Solve_modular_double (
    const double p,
    const size_t M,
    double * A,
    const size_t lda,
    double * x,
    const int incx,
    const double * b,
    const int incb,
    bool positive)
```

**13.233.2.56 solveLB\_modular\_double()**

```
void solveLB_modular_double (
    const double p,
    const enum FFLAS::FFLAS_SIDE Side,
    const size_t M,
    const size_t N,
    const size_t R,
    double * L,
    const size_t ldl,
    const size_t * Q,
    double * B,
    const size_t ldb,
    bool positive)
```

**13.233.2.57 solveLB2\_modular\_double()**

```
void solveLB2_modular_double (
    const double p,
    const enum FFLAS::FFLAS_SIDE Side,
    const size_t M,
    const size_t N,
    const size_t R,
    double * L,
    const size_t ldl,
    const size_t * Q,
    double * B,
    const size_t ldb,
    bool positive)
```

**13.233.2.58 RandomNullSpaceVector\_modular\_double()**

```
void RandomNullSpaceVector_modular_double (
    const double p,
    const enum FFLAS::FFLAS_SIDE Side,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    double * X,
    const size_t incX,
    bool positive)
```

**13.233.2.59 NullSpaceBasis\_modular\_double()**

```
size_t NullSpaceBasis_modular_double (
    const double p,
```

```

const enum FFLAS::FFLAS_SIDE Side,
const size_t M,
const size_t N,
double * A,
const size_t lda,
double ** NS,
size_t * ldn,
size_t * NSdim,
bool positive)

```

#### 13.233.2.60 RowRankProfile\_modular\_double()

```

size_t RowRankProfile_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t ** rkprofile,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

#### 13.233.2.61 ColumnRankProfile\_modular\_double()

```

size_t ColumnRankProfile_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t ** rkprofile,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

#### 13.233.2.62 RankProfileFromLU()

```

void RankProfileFromLU (
    const size_t * P,
    const size_t N,
    const size_t R,
    size_t * rkprofile,
    const enum FFPACK_C_LU_TAG LuTag)

```

#### 13.233.2.63 LeadingSubmatrixRankProfiles()

```

size_t LeadingSubmatrixRankProfiles (
    const size_t M,
    const size_t N,
    const size_t R,
    const size_t LSm,
    const size_t LSn,
    const size_t * P,
    const size_t * Q,
    size_t * RRP,
    size_t * CRP)

```

#### 13.233.2.64 RowRankProfileSubmatrixIndices\_modular\_double()

```

size_t RowRankProfileSubmatrixIndices_modular_double (

```

```

    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t ** rowindices,
    size_t ** colindices,
    size_t * R,
    bool positive)

```

### 13.233.2.65 ColRankProfileSubmatrixIndices\_modular\_double()

```

size_t ColRankProfileSubmatrixIndices_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t ** rowindices,
    size_t ** colindices,
    size_t * R,
    bool positive)

```

### 13.233.2.66 RowRankProfileSubmatrix\_modular\_double()

```

size_t RowRankProfileSubmatrix_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    double ** X,
    size_t * R,
    bool positive)

```

### 13.233.2.67 ColRankProfileSubmatrix\_modular\_double()

```

size_t ColRankProfileSubmatrix_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    double ** X,
    size_t * R,
    bool positive)

```

### 13.233.2.68 getTriangular\_modular\_double()

```

void getTriangular_modular_double (
    const double p,
    const enum FFLAS::FFLAS_UPLO Uplo,
    const enum FFLAS::FFLAS_DIAG Diag,
    const size_t M,
    const size_t N,
    const size_t R,
    const double * A,
    const size_t lda,
    double * T,

```



```
const size_t ldt,  
const bool OnlyNonZeroVectors,  
bool positive)
```

### 13.233.2.69 getTriangularin\_modular\_double()

```
void getTriangularin_modular_double (  
    const double p,  
    const enum FFLAS::FFLAS_UPLO Uplo,  
    const enum FFLAS::FFLAS_DIAG Diag,  
    const size_t M,  
    const size_t N,  
    const size_t R,  
    double * A,  
    const size_t lda,  
    bool positive)
```

### 13.233.2.70 getEchelonForm\_modular\_double()

```
void getEchelonForm_modular_double (  
    const double p,  
    const enum FFLAS::FFLAS_UPLO Uplo,  
    const enum FFLAS::FFLAS_DIAG Diag,  
    const size_t M,  
    const size_t N,  
    const size_t R,  
    const size_t * P,  
    const double * A,  
    const size_t lda,  
    double * T,  
    const size_t ldt,  
    const bool OnlyNonZeroVectors,  
    const enum FFPACK_C_LU_TAG LuTag,  
    bool positive)
```

### 13.233.2.71 getEchelonFormin\_modular\_double()

```
void getEchelonFormin_modular_double (  
    const double p,  
    const enum FFLAS::FFLAS_UPLO Uplo,  
    const enum FFLAS::FFLAS_DIAG Diag,  
    const size_t M,  
    const size_t N,  
    const size_t R,  
    const size_t * P,  
    double * A,  
    const size_t lda,  
    const enum FFPACK_C_LU_TAG LuTag,  
    bool positive)
```

### 13.233.2.72 getEchelonTransform\_modular\_double()

```
void getEchelonTransform_modular_double (  
    const double p,  
    const enum FFLAS::FFLAS_UPLO Uplo,  
    const enum FFLAS::FFLAS_DIAG Diag,  
    const size_t M,  
    const size_t N,  
    const size_t R,
```

```

    const size_t * P,
    const size_t * Q,
    const double * A,
    const size_t lda,
    double * T,
    const size_t ldt,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

### 13.233.2.73 getReducedEchelonForm\_modular\_double()

```

void getReducedEchelonForm_modular_double (
    const double p,
    const enum FFLAS::FFLAS_UPLO Uplo,
    const size_t M,
    const size_t N,
    const size_t R,
    const size_t * P,
    const double * A,
    const size_t lda,
    double * T,
    const size_t ldt,
    const bool OnlyNonZeroVectors,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

### 13.233.2.74 getReducedEchelonFormin\_modular\_double()

```

void getReducedEchelonFormin_modular_double (
    const double p,
    const enum FFLAS::FFLAS_UPLO Uplo,
    const size_t M,
    const size_t N,
    const size_t R,
    const size_t * P,
    double * A,
    const size_t lda,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

### 13.233.2.75 getReducedEchelonTransform\_modular\_double()

```

void getReducedEchelonTransform_modular_double (
    const double p,
    const enum FFLAS::FFLAS_UPLO Uplo,
    const size_t M,
    const size_t N,
    const size_t R,
    const size_t * P,
    const size_t * Q,
    const double * A,
    const size_t lda,
    double * T,
    const size_t ldt,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

**13.233.2.76 PLUQtoEchelonPermutation()**

```
void PLUQtoEchelonPermutation (
    const size_t N,
    const size_t R,
    const size_t * P,
    size_t * outPerm)
```

**13.234 ffpack\_c.h File Reference**

```
#include <stdbool.h>
#include <stdlib.h>
#include <inttypes.h>
```

**Macros**

- `#define FFPACK_COMPILED`

**Enumerations**

- enum `FFLAS_C_ORDER` { `FflasRowMajor` =101 , `FflasColMajor` =102 }
- enum `FFLAS_C_TRANSPOSE` { `FflasNoTrans` = 111 , `FflasTrans` = 112 }
- enum `FFLAS_C_UPLO` { `FflasUpper` = 121 , `FflasLower` = 122 }
- enum `FFLAS_C_DIAG` { `FflasNonUnit` = 131 , `FflasUnit` = 132 }
- enum `FFLAS_C_SIDE` { `FflasLeft` = 141 , `FflasRight` = 142 }
- enum `FFPACK_C_LU_TAG` { `FfpackSlabRecursive` = 1 , `FfpackTileRecursive` = 2 , `FfpackSingular` = 3 }
- enum `FFPACK_C_CHARPOLY_TAG` {  
`FfpackLUK` =1 , `FfpackKG` =2 , `FfpackHybrid` =3 , `FfpackKGFast` =4 ,  
`FfpackDanilevski` =5 , `FfpackArithProg` =6 , `FfpackKGFastG` =7 }
- enum `FFPACK_C_MINPOLY_TAG` { `FfpackDense` =1 , `FfpackKGF` =2 }

**Functions**

- void `LAPACKPerm2MathPerm` (size\_t \*MathP, const size\_t \*LapackP, const size\_t N)
- void `MathPerm2LAPACKPerm` (size\_t \*LapackP, const size\_t \*MathP, const size\_t N)
- void `MatrixApplyS_modular_double` (const double p, double \*A, const size\_t lda, const size\_t width, const size\_t M2, const size\_t R1, const size\_t R2, const size\_t R3, const size\_t R4, bool positive)
- void `PermApplyS_double` (double \*A, const size\_t lda, const size\_t width, const size\_t M2, const size\_t R1, const size\_t R2, const size\_t R3, const size\_t R4)
- void `MatrixApplyT_modular_double` (const double p, double \*A, const size\_t lda, const size\_t width, const size\_t N2, const size\_t R1, const size\_t R2, const size\_t R3, const size\_t R4, bool positive)
- void `PermApplyT_double` (double \*A, const size\_t lda, const size\_t width, const size\_t N2, const size\_t R1, const size\_t R2, const size\_t R3, const size\_t R4)
- void `composePermutationsLLM` (size\_t \*MathP, const size\_t \*P1, const size\_t \*P2, const size\_t R, const size\_t N)
- void `composePermutationsLLL` (size\_t \*P1, const size\_t \*P2, const size\_t R, const size\_t N)
- void `composePermutationsMLM` (size\_t \*MathP1, const size\_t \*P2, const size\_t R, const size\_t N)
- void `cyclic_shift_mathPerm` (size\_t \*P, const size\_t s)
- void `cyclic_shift_row_modular_double` (const double p, double \*A, size\_t m, size\_t n, size\_t lda, bool positive)
- void `cyclic_shift_col_modular_double` (const double p, double \*A, size\_t m, size\_t n, size\_t lda, bool positive)
- void `applyP_modular_double` (const double p, const enum `FFLAS_C_SIDE` Side, const enum `FFLAS_C_TRANSPOSE` Trans, const size\_t M, const size\_t ibeg, const size\_t iend, double \*A, const size\_t lda, const size\_t \*P, bool positive)
- void `fgetsrsin_modular_double` (const double p, const enum `FFLAS_C_SIDE` Side, const size\_t M, const size\_t N, const size\_t R, double \*A, const size\_t lda, const size\_t \*P, const size\_t \*Q, double \*B, const size\_t ldb, int \*info, bool positive)

- `double * fgetrs_modular_double` (const double p, const enum `FFLAS_C_SIDE` Side, const size\_t M, const size\_t N, const size\_t NRHS, const size\_t R, double \*A, const size\_t lda, const size\_t \*P, const size\_t \*Q, double \*X, const size\_t ldx, const double \*B, const size\_t ldb, int \*info, bool positive)
- `size_t fgesvin_modular_double` (const double p, const enum `FFLAS_C_SIDE` Side, const size\_t M, const size\_t N, double \*A, const size\_t lda, double \*B, const size\_t ldb, int \*info, bool positive)
- `size_t fgesv_modular_double` (const double p, const enum `FFLAS_C_SIDE` Side, const size\_t M, const size\_t N, const size\_t NRHS, double \*A, const size\_t lda, double \*X, const size\_t ldx, const double \*B, const size\_t ldb, int \*info)
- `void ftrtri_modular_double` (const double p, const enum `FFLAS_C_UPLO` Uplo, const enum `FFLAS_C_DIAG` Diag, const size\_t N, double \*A, const size\_t lda, bool positive)
- `void trinv_left_modular_double` (const double p, const size\_t N, const double \*L, const size\_t ldl, double \*X, const size\_t ldx, bool positive)
- `void ftrrm_modular_double` (const double p, const enum `FFLAS_C_DIAG` diag, const size\_t N, double \*A, const size\_t lda, bool positive)
- `size_t PLUQ_modular_double` (const double p, const enum `FFLAS_C_DIAG` Diag, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Q, bool positive)
- `size_t LUdivine_modular_double` (const double p, const enum `FFLAS_C_DIAG` Diag, const enum `FFLAS_C_TRANSPOSE` trans, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const enum `FFPACK_C_LU_TAG` LuTag, const size\_t cutoff, bool positive)
- `size_t LUdivine_small_modular_double` (const double p, const enum `FFLAS_C_DIAG` Diag, const enum `FFLAS_C_TRANSPOSE` trans, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Q, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t LUdivine_gauss_modular_double` (const double p, const enum `FFLAS_C_DIAG` Diag, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Q, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t ColumnEchelonForm_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t RowEchelonForm_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t ColumnEchelonForm_modular_float` (const float p, const size\_t M, const size\_t N, float \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t RowEchelonForm_modular_float` (const float p, const size\_t M, const size\_t N, float \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t ColumnEchelonForm_modular_int32_t` (const int32\_t p, const size\_t M, const size\_t N, int32\_t \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t RowEchelonForm_modular_int32_t` (const int32\_t p, const size\_t M, const size\_t N, int32\_t \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t ReducedColumnEchelonForm_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t ReducedRowEchelonForm_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t ReducedColumnEchelonForm_modular_float` (const float p, const size\_t M, const size\_t N, float \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t ReducedRowEchelonForm_modular_float` (const float p, const size\_t M, const size\_t N, float \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t ReducedColumnEchelonForm_modular_int32_t` (const int32\_t p, const size\_t M, const size\_t N, int32\_t \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t ReducedRowEchelonForm_modular_int32_t` (const int32\_t p, const size\_t M, const size\_t N, int32\_t \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t ReducedRowEchelonForm2_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, bool positive)

- `size_t REF_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, const size\_t colbeg, const size\_t rowbeg, const size\_t colsize, size\_t \*Qt, size\_t \*P, bool positive)
- `double * Invertin_modular_double` (const double p, const size\_t M, double \*A, const size\_t lda, int \*nullity, bool positive)
- `double * Invert_modular_double` (const double p, const size\_t M, const double \*A, const size\_t lda, double \*X, const size\_t ldx, int \*nullity, bool positive)
- `double * Invert2_modular_double` (const double p, const size\_t M, double \*A, const size\_t lda, double \*X, const size\_t ldx, int \*nullity, bool positive)
- `size_t KrylovElim_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Q, const size\_t deg, size\_t \*iterates, size\_t \*inviterates, const size\_t maxit, size\_t virt, bool positive)
- `size_t SpecRankProfile_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, const size\_t deg, size\_t \*rankProfile, bool positive)
- `size_t Rank_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, bool positive)
- `bool IsSingular_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, bool positive)
- `double Det_modular_double` (const double p, const size\_t N, double \*A, const size\_t lda, bool positive)
- `double * Solve_modular_double` (const double p, const size\_t M, double \*A, const size\_t lda, double \*x, const int incx, const double \*b, const int incb, bool positive)
- `void solveLB_modular_double` (const double p, const enum FFLAS\_C\_SIDE Side, const size\_t M, const size\_t N, const size\_t R, double \*L, const size\_t ldl, const size\_t \*Q, double \*B, const size\_t ldb)
- `void solveLB2_modular_double` (const double p, const enum FFLAS\_C\_SIDE Side, const size\_t M, const size\_t N, const size\_t R, double \*L, const size\_t ldl, const size\_t \*Q, double \*B, const size\_t ldb, bool positive)
- `void RandomNullSpaceVector_modular_double` (const double p, const enum FFLAS\_C\_SIDE Side, const size\_t M, const size\_t N, double \*A, const size\_t lda, double \*X, const size\_t incX, bool positive)
- `size_t NullSpaceBasis_modular_double` (const double p, const enum FFLAS\_C\_SIDE Side, const size\_t M, const size\_t N, double \*A, const size\_t lda, double \*\*NS, size\_t \*ldn, size\_t \*NSdim, bool positive)
- `size_t RowRankProfile_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*\*rkprofile, const enum FFPACK\_C\_LU\_TAG LuTag, bool positive)
- `size_t ColumnRankProfile_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*\*rkprofile, const enum FFPACK\_C\_LU\_TAG LuTag, bool positive)
- `void RankProfileFromLU` (const size\_t \*P, const size\_t N, const size\_t R, size\_t \*rkprofile, const enum FFPACK\_C\_LU\_TAG LuTag)
- `size_t LeadingSubmatrixRankProfiles` (const size\_t M, const size\_t N, const size\_t R, const size\_t LSm, const size\_t LSn, const size\_t \*P, const size\_t \*Q, size\_t \*RRP, size\_t \*CRP)
- `size_t RowRankProfileSubmatrixIndices_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*\*rowindices, size\_t \*\*colindices, size\_t \*R, bool positive)
- `size_t ColRankProfileSubmatrixIndices_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*\*rowindices, size\_t \*\*colindices, size\_t \*R, bool positive)
- `size_t RowRankProfileSubmatrix_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, double \*\*X, size\_t \*R, bool positive)
- `size_t ColRankProfileSubmatrix_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, double \*\*X, size\_t \*R, bool positive)
- `void getTriangular_modular_double` (const double p, const enum FFLAS\_C\_UPLO Uplo, const enum FFLAS\_C\_DIAG diag, const size\_t M, const size\_t N, const size\_t R, const double \*A, const size\_t lda, double \*T, const size\_t ldt, const bool OnlyNonZeroVectors, bool positive)
- `void getTriangularin_modular_double` (const double p, const enum FFLAS\_C\_UPLO Uplo, const enum FFLAS\_C\_DIAG diag, const size\_t M, const size\_t N, const size\_t R, double \*A, const size\_t lda, bool positive)
- `void getEchelonForm_modular_double` (const double p, const enum FFLAS\_C\_UPLO Uplo, const enum FFLAS\_C\_DIAG diag, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const double \*A, const size\_t lda, double \*T, const size\_t ldt, const bool OnlyNonZeroVectors, const enum FFPACK\_C\_LU\_TAG LuTag, bool positive)
- `void getEchelonFormin_modular_double` (const double p, const enum FFLAS\_C\_UPLO Uplo, const enum FFLAS\_C\_DIAG diag, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, double \*A, const size\_t lda, const enum FFPACK\_C\_LU\_TAG LuTag, bool positive)

- void [getEchelonTransform\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const enum [FFLAS\\_C\\_DIAG](#) diag, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const size\_t \*Q, const double \*A, const size\_t lda, double \*T, const size\_t ldt, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- void [getReducedEchelonForm\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const double \*A, const size\_t lda, double \*T, const size\_t ldt, const bool OnlyNonZeroVectors, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- void [getReducedEchelonFormin\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, double \*A, const size\_t lda, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- void [getReducedEchelonTransform\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const size\_t \*Q, const double \*A, const size\_t lda, double \*T, const size\_t ldt, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- void [PLUQtoEchelonPermutation](#) (const size\_t N, const size\_t R, const size\_t \*P, size\_t \*outPerm)

## 13.234.1 Macro Definition Documentation

### 13.234.1.1 FFPACK\_COMPILED

```
#define FFPACK_COMPILED
```

## 13.234.2 Enumeration Type Documentation

### 13.234.2.1 FFLAS\_C\_ORDER

```
enum FFLAS\_C\_ORDER
```

Enumerator

|               |  |
|---------------|--|
| FflasRowMajor |  |
| FflasColMajor |  |

### 13.234.2.2 FFLAS\_C\_TRANSPOSE

```
enum FFLAS\_C\_TRANSPOSE
```

Enumerator

|              |  |
|--------------|--|
| FflasNoTrans |  |
| FflasTrans   |  |

### 13.234.2.3 FFLAS\_C\_UPLO

```
enum FFLAS\_C\_UPLO
```

Enumerator

|            |  |
|------------|--|
| FflasUpper |  |
| FflasLower |  |

### 13.234.2.4 FFLAS\_C\_DIAG

```
enum FFLAS\_C\_DIAG
```

Enumerator

|              |  |
|--------------|--|
| FflasNonUnit |  |
| FflasUnit    |  |

### 13.234.2.5 FFLAS\_C\_SIDE

enum [FFLAS\\_C\\_SIDE](#)

#### Enumerator

|            |  |
|------------|--|
| FflasLeft  |  |
| FflasRight |  |

### 13.234.2.6 FFPACK\_C\_LU\_TAG

enum [FFPACK\\_C\\_LU\\_TAG](#)

#### Enumerator

|                     |  |
|---------------------|--|
| FfpackSlabRecursive |  |
| FfpackTileRecursive |  |
| FfpackSingular      |  |

### 13.234.2.7 FFPACK\_C\_CHARPOLY\_TAG

enum [FFPACK\\_C\\_CHARPOLY\\_TAG](#)

#### Enumerator

|                  |  |
|------------------|--|
| FfpackLUK        |  |
| FfpackKG         |  |
| FfpackHybrid     |  |
| FfpackKGFast     |  |
| FfpackDanilevski |  |
| FfpackArithProg  |  |
| FfpackKGFastG    |  |

### 13.234.2.8 FFPACK\_C\_MINPOLY\_TAG

enum [FFPACK\\_C\\_MINPOLY\\_TAG](#)

#### Enumerator

|             |  |
|-------------|--|
| FfpackDense |  |
| FfpackKGF   |  |

## 13.234.3 Function Documentation

### 13.234.3.1 LAPACKPerm2MathPerm()

```
void LAPACKPerm2MathPerm (  
    size_t * MathP,  
    const size_t * LapackP,  
    const size_t N)
```

### 13.234.3.2 MathPerm2LAPACKPerm()

```
void MathPerm2LAPACKPerm (
    size_t * LapackP,
    const size_t * MathP,
    const size_t N)
```

### 13.234.3.3 MatrixApplyS\_modular\_double()

```
void MatrixApplyS_modular_double (
    const double p,
    double * A,
    const size_t lda,
    const size_t width,
    const size_t M2,
    const size_t R1,
    const size_t R2,
    const size_t R3,
    const size_t R4,
    bool positive)
```

### 13.234.3.4 PermApplyS\_double()

```
void PermApplyS_double (
    double * A,
    const size_t lda,
    const size_t width,
    const size_t M2,
    const size_t R1,
    const size_t R2,
    const size_t R3,
    const size_t R4)
```

### 13.234.3.5 MatrixApplyT\_modular\_double()

```
void MatrixApplyT_modular_double (
    const double p,
    double * A,
    const size_t lda,
    const size_t width,
    const size_t N2,
    const size_t R1,
    const size_t R2,
    const size_t R3,
    const size_t R4,
    bool positive)
```

### 13.234.3.6 PermApplyT\_double()

```
void PermApplyT_double (
    double * A,
    const size_t lda,
    const size_t width,
    const size_t N2,
    const size_t R1,
    const size_t R2,
    const size_t R3,
    const size_t R4)
```



**13.234.3.7 composePermutationsLLM()**

```
void composePermutationsLLM (
    size_t * MathP,
    const size_t * P1,
    const size_t * P2,
    const size_t R,
    const size_t N)
```

**13.234.3.8 composePermutationsLLL()**

```
void composePermutationsLLL (
    size_t * P1,
    const size_t * P2,
    const size_t R,
    const size_t N)
```

**13.234.3.9 composePermutationsMLM()**

```
void composePermutationsMLM (
    size_t * MathP1,
    const size_t * P2,
    const size_t R,
    const size_t N)
```

**13.234.3.10 cyclic\_shift\_mathPerm()**

```
void cyclic_shift_mathPerm (
    size_t * P,
    const size_t s)
```

**13.234.3.11 cyclic\_shift\_row\_modular\_double()**

```
void cyclic_shift_row_modular_double (
    const double p,
    double * A,
    size_t m,
    size_t n,
    size_t lda,
    bool positive)
```

**13.234.3.12 cyclic\_shift\_col\_modular\_double()**

```
void cyclic_shift_col_modular_double (
    const double p,
    double * A,
    size_t m,
    size_t n,
    size_t lda,
    bool positive)
```

**13.234.3.13 applyP\_modular\_double()**

```
void applyP_modular_double (
    const double p,
    const enum FFLAS_C_SIDE Side,
    const enum FFLAS_C_TRANSPOSE Trans,
    const size_t M,
    const size_t ibeg,
    const size_t iend,
```

```
double * A,
const size_t lda,
const size_t * P,
bool positive)
```

#### 13.234.3.14 fgetrsin\_modular\_double()

```
void fgetrsin_modular_double (
    const double p,
    const enum FFLAS_C_SIDE Side,
    const size_t M,
    const size_t N,
    const size_t R,
    double * A,
    const size_t lda,
    const size_t * P,
    const size_t * Q,
    double * B,
    const size_t ldb,
    int * info,
    bool positive)
```

#### 13.234.3.15 fgetrs\_modular\_double()

```
double * fgetrs_modular_double (
    const double p,
    const enum FFLAS_C_SIDE Side,
    const size_t M,
    const size_t N,
    const size_t NRHS,
    const size_t R,
    double * A,
    const size_t lda,
    const size_t * P,
    const size_t * Q,
    double * X,
    const size_t ldx,
    const double * B,
    const size_t ldb,
    int * info,
    bool positive)
```

#### 13.234.3.16 fgesvin\_modular\_double()

```
size_t fgesvin_modular_double (
    const double p,
    const enum FFLAS_C_SIDE Side,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    double * B,
    const size_t ldb,
    int * info,
    bool positive)
```

#### 13.234.3.17 fgesv\_modular\_double()

```
size_t fgesv_modular_double (
```

```
const double p,
const enum FFLAS_C_SIDE Side,
const size_t M,
const size_t N,
const size_t NRHS,
double * A,
const size_t lda,
double * X,
const size_t ldx,
const double * B,
const size_t ldb,
int * info)
```

#### 13.234.3.18 ftrtri\_modular\_double()

```
void ftrtri_modular_double (
    const double p,
    const enum FFLAS_C_UPLO Uplo,
    const enum FFLAS_C_DIAG Diag,
    const size_t N,
    double * A,
    const size_t lda,
    bool positive)
```

#### 13.234.3.19 trinv\_left\_modular\_double()

```
void trinv_left_modular_double (
    const double p,
    const size_t N,
    const double * L,
    const size_t ldl,
    double * X,
    const size_t ldx,
    bool positive)
```

#### 13.234.3.20 ftrtrm\_modular\_double()

```
void ftrtrm_modular_double (
    const double p,
    const enum FFLAS_C_DIAG diag,
    const size_t N,
    double * A,
    const size_t lda,
    bool positive)
```

#### 13.234.3.21 PLUQ\_modular\_double()

```
size_t PLUQ_modular_double (
    const double p,
    const enum FFLAS_C_DIAG Diag,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t * P,
    size_t * Q,
    bool positive)
```

**13.234.3.22 LUdivine\_modular\_double()**

```

size_t LUdivine_modular_double (
    const double p,
    const enum FFLAS_C_DIAG Diag,
    const enum FFLAS_C_TRANSPOSE trans,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const enum FFPACK_C_LU_TAG LuTag,
    const size_t cutoff,
    bool positive)

```

**13.234.3.23 LUdivine\_small\_modular\_double()**

```

size_t LUdivine_small_modular_double (
    const double p,
    const enum FFLAS_C_DIAG Diag,
    const enum FFLAS_C_TRANSPOSE trans,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t * P,
    size_t * Q,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

**13.234.3.24 LUdivine\_gauss\_modular\_double()**

```

size_t LUdivine_gauss_modular_double (
    const double p,
    const enum FFLAS_C_DIAG Diag,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t * P,
    size_t * Q,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

**13.234.3.25 ColumnEchelonForm\_modular\_double()**

```

size_t ColumnEchelonForm_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

**13.234.3.26 RowEchelonForm\_modular\_double()**

```
size_t RowEchelonForm_modular_double (  
    const double p,  
    const size_t M,  
    const size_t N,  
    double * A,  
    const size_t lda,  
    size_t * P,  
    size_t * Qt,  
    const bool transform,  
    const enum FFPACK_C_LU_TAG LuTag,  
    bool positive)
```

**13.234.3.27 ColumnEchelonForm\_modular\_float()**

```
size_t ColumnEchelonForm_modular_float (  
    const float p,  
    const size_t M,  
    const size_t N,  
    float * A,  
    const size_t lda,  
    size_t * P,  
    size_t * Qt,  
    bool transform,  
    const enum FFPACK_C_LU_TAG LuTag,  
    bool positive)
```

**13.234.3.28 RowEchelonForm\_modular\_float()**

```
size_t RowEchelonForm_modular_float (  
    const float p,  
    const size_t M,  
    const size_t N,  
    float * A,  
    const size_t lda,  
    size_t * P,  
    size_t * Qt,  
    const bool transform,  
    const enum FFPACK_C_LU_TAG LuTag,  
    bool positive)
```

**13.234.3.29 ColumnEchelonForm\_modular\_int32\_t()**

```
size_t ColumnEchelonForm_modular_int32_t (  
    const int32_t p,  
    const size_t M,  
    const size_t N,  
    int32_t * A,  
    const size_t lda,  
    size_t * P,  
    size_t * Qt,  
    bool transform,  
    const enum FFPACK_C_LU_TAG LuTag,  
    bool positive)
```

**13.234.3.30 RowEchelonForm\_modular\_int32\_t()**

```
size_t RowEchelonForm_modular_int32_t (  
    const int32_t p,
```

```

    const size_t M,
    const size_t N,
    int32_t * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

### 13.234.3.31 ReducedColumnEchelonForm\_modular\_double()

```

size_t ReducedColumnEchelonForm_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

### 13.234.3.32 ReducedRowEchelonForm\_modular\_double()

```

size_t ReducedRowEchelonForm_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

### 13.234.3.33 ReducedColumnEchelonForm\_modular\_float()

```

size_t ReducedColumnEchelonForm_modular_float (
    const float p,
    const size_t M,
    const size_t N,
    float * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

### 13.234.3.34 ReducedRowEchelonForm\_modular\_float()

```

size_t ReducedRowEchelonForm_modular_float (
    const float p,
    const size_t M,
    const size_t N,
    float * A,

```

```

    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

#### 13.234.3.35 ReducedColumnEchelonForm\_modular\_int32\_t()

```

size_t ReducedColumnEchelonForm_modular_int32_t (
    const int32_t p,
    const size_t M,
    const size_t N,
    int32_t * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

#### 13.234.3.36 ReducedRowEchelonForm\_modular\_int32\_t()

```

size_t ReducedRowEchelonForm_modular_int32_t (
    const int32_t p,
    const size_t M,
    const size_t N,
    int32_t * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

#### 13.234.3.37 ReducedRowEchelonForm2\_modular\_double()

```

size_t ReducedRowEchelonForm2_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t * P,
    size_t * Qt,
    const bool transform,
    bool positive)

```

#### 13.234.3.38 REF\_modular\_double()

```

size_t REF_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    const size_t colbeg,
    const size_t rowbeg,
    const size_t colsize,

```

```

    size_t * Qt,
    size_t * P,
    bool positive)

```

#### 13.234.3.39 Invertin\_modular\_double()

```

double * Invertin_modular_double (
    const double p,
    const size_t M,
    double * A,
    const size_t lda,
    int * nullity,
    bool positive)

```

#### 13.234.3.40 Invert\_modular\_double()

```

double * Invert_modular_double (
    const double p,
    const size_t M,
    const double * A,
    const size_t lda,
    double * X,
    const size_t ldx,
    int * nullity,
    bool positive)

```

#### 13.234.3.41 Invert2\_modular\_double()

```

double * Invert2_modular_double (
    const double p,
    const size_t M,
    double * A,
    const size_t lda,
    double * X,
    const size_t ldx,
    int * nullity,
    bool positive)

```

#### 13.234.3.42 KrylovElim\_modular\_double()

```

size_t KrylovElim_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t * P,
    size_t * Q,
    const size_t deg,
    size_t * iterates,
    size_t * inviterates,
    const size_t maxit,
    size_t virt,
    bool positive)

```

#### 13.234.3.43 SpecRankProfile\_modular\_double()

```

size_t SpecRankProfile_modular_double (
    const double p,

```



```
    const size_t M,  
    const size_t N,  
    double * A,  
    const size_t lda,  
    const size_t deg,  
    size_t * rankProfile,  
    bool positive)
```

#### 13.234.3.44 Rank\_modular\_double()

```
size_t Rank_modular_double (  
    const double p,  
    const size_t M,  
    const size_t N,  
    double * A,  
    const size_t lda,  
    bool positive)
```

#### 13.234.3.45 IsSingular\_modular\_double()

```
bool IsSingular_modular_double (  
    const double p,  
    const size_t M,  
    const size_t N,  
    double * A,  
    const size_t lda,  
    bool positive)
```

#### 13.234.3.46 Det\_modular\_double()

```
double Det_modular_double (  
    const double p,  
    const size_t N,  
    double * A,  
    const size_t lda,  
    bool positive)
```

#### 13.234.3.47 Solve\_modular\_double()

```
double * Solve_modular_double (  
    const double p,  
    const size_t M,  
    double * A,  
    const size_t lda,  
    double * x,  
    const int incx,  
    const double * b,  
    const int incb,  
    bool positive)
```

#### 13.234.3.48 solveLB\_modular\_double()

```
void solveLB_modular_double (  
    const double p,  
    const enum FFLAS_C_SIDE Side,  
    const size_t M,  
    const size_t N,  
    const size_t R,  
    double * L,
```

```

    const size_t ldl,
    const size_t * Q,
    double * B,
    const size_t ldb)

```

#### 13.234.3.49 solveLB2\_modular\_double()

```

void solveLB2_modular_double (
    const double p,
    const enum FFLAS_C_SIDE Side,
    const size_t M,
    const size_t N,
    const size_t R,
    double * L,
    const size_t ldl,
    const size_t * Q,
    double * B,
    const size_t ldb,
    bool positive)

```

#### 13.234.3.50 RandomNullSpaceVector\_modular\_double()

```

void RandomNullSpaceVector_modular_double (
    const double p,
    const enum FFLAS_C_SIDE Side,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    double * X,
    const size_t incX,
    bool positive)

```

#### 13.234.3.51 NullSpaceBasis\_modular\_double()

```

size_t NullSpaceBasis_modular_double (
    const double p,
    const enum FFLAS_C_SIDE Side,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    double ** NS,
    size_t * ldn,
    size_t * NSdim,
    bool positive)

```

#### 13.234.3.52 RowRankProfile\_modular\_double()

```

size_t RowRankProfile_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t ** rkprofile,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

**13.234.3.53 ColumnRankProfile\_modular\_double()**

```
size_t ColumnRankProfile_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t ** rkprofile,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)
```

**13.234.3.54 RankProfileFromLU()**

```
void RankProfileFromLU (
    const size_t * P,
    const size_t N,
    const size_t R,
    size_t * rkprofile,
    const enum FFPACK_C_LU_TAG LuTag)
```

**13.234.3.55 LeadingSubmatrixRankProfiles()**

```
size_t LeadingSubmatrixRankProfiles (
    const size_t M,
    const size_t N,
    const size_t R,
    const size_t LSm,
    const size_t LSn,
    const size_t * P,
    const size_t * Q,
    size_t * RRP,
    size_t * CRP)
```

**13.234.3.56 RowRankProfileSubmatrixIndices\_modular\_double()**

```
size_t RowRankProfileSubmatrixIndices_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t ** rowindices,
    size_t ** colindices,
    size_t * R,
    bool positive)
```

**13.234.3.57 ColRankProfileSubmatrixIndices\_modular\_double()**

```
size_t ColRankProfileSubmatrixIndices_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    size_t ** rowindices,
    size_t ** colindices,
    size_t * R,
    bool positive)
```

**13.234.3.58 RowRankProfileSubmatrix\_modular\_double()**

```
size_t RowRankProfileSubmatrix_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    double ** X,
    size_t * R,
    bool positive)
```

**13.234.3.59 ColRankProfileSubmatrix\_modular\_double()**

```
size_t ColRankProfileSubmatrix_modular_double (
    const double p,
    const size_t M,
    const size_t N,
    double * A,
    const size_t lda,
    double ** X,
    size_t * R,
    bool positive)
```

**13.234.3.60 getTriangular\_modular\_double()**

```
void getTriangular_modular_double (
    const double p,
    const enum FFLAS_C_UPLO Uplo,
    const enum FFLAS_C_DIAG diag,
    const size_t M,
    const size_t N,
    const size_t R,
    const double * A,
    const size_t lda,
    double * T,
    const size_t ldt,
    const bool OnlyNonZeroVectors,
    bool positive)
```

**13.234.3.61 getTriangularin\_modular\_double()**

```
void getTriangularin_modular_double (
    const double p,
    const enum FFLAS_C_UPLO Uplo,
    const enum FFLAS_C_DIAG diag,
    const size_t M,
    const size_t N,
    const size_t R,
    double * A,
    const size_t lda,
    bool positive)
```

**13.234.3.62 getEchelonForm\_modular\_double()**

```
void getEchelonForm_modular_double (
    const double p,
    const enum FFLAS_C_UPLO Uplo,
    const enum FFLAS_C_DIAG diag,
```

```

    const size_t M,
    const size_t N,
    const size_t R,
    const size_t * P,
    const double * A,
    const size_t lda,
    double * T,
    const size_t ldt,
    const bool OnlyNonZeroVectors,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

### 13.234.3.63 getEchelonFormin\_modular\_double()

```

void getEchelonFormin_modular_double (
    const double p,
    const enum FFLAS_C_UPLO Uplo,
    const enum FFLAS_C_DIAG diag,
    const size_t M,
    const size_t N,
    const size_t R,
    const size_t * P,
    double * A,
    const size_t lda,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

### 13.234.3.64 getEchelonTransform\_modular\_double()

```

void getEchelonTransform_modular_double (
    const double p,
    const enum FFLAS_C_UPLO Uplo,
    const enum FFLAS_C_DIAG diag,
    const size_t M,
    const size_t N,
    const size_t R,
    const size_t * P,
    const size_t * Q,
    const double * A,
    const size_t lda,
    double * T,
    const size_t ldt,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)

```

### 13.234.3.65 getReducedEchelonForm\_modular\_double()

```

void getReducedEchelonForm_modular_double (
    const double p,
    const enum FFLAS_C_UPLO Uplo,
    const size_t M,
    const size_t N,
    const size_t R,
    const size_t * P,
    const double * A,
    const size_t lda,
    double * T,
    const size_t ldt,
    const bool OnlyNonZeroVectors,

```

```
const enum FFPACK_C_LU_TAG LuTag,
bool positive)
```

### 13.234.3.66 getReducedEchelonFormin\_modular\_double()

```
void getReducedEchelonFormin_modular_double (
    const double p,
    const enum FFLAS_C_UPLO Uplo,
    const size_t M,
    const size_t N,
    const size_t R,
    const size_t * P,
    double * A,
    const size_t lda,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)
```

### 13.234.3.67 getReducedEchelonTransform\_modular\_double()

```
void getReducedEchelonTransform_modular_double (
    const double p,
    const enum FFLAS_C_UPLO Uplo,
    const size_t M,
    const size_t N,
    const size_t R,
    const size_t * P,
    const size_t * Q,
    const double * A,
    const size_t lda,
    double * T,
    const size_t ldt,
    const enum FFPACK_C_LU_TAG LuTag,
    bool positive)
```

### 13.234.3.68 PLUQtoEchelonPermutation()

```
void PLUQtoEchelonPermutation (
    const size_t N,
    const size_t R,
    const size_t * P,
    size_t * outPerm)
```

## 13.235 ffpack\_inst.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "givaro/modular.h"
#include "givaro/modular-balanced.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "ffpack_inst_implem.inl"
```

### Macros

- #define \_\_FFPACK\_INST\_C
- #define FFLAS\_COMPILED
- #define INST\_OR\_DECL
- #define FFLAS\_FIELD Givaro::ModularBalanced
- #define FFLAS\_ELT double

- `#define FFLAS_ELT float`
- `#define FFLAS_ELT int64_t`
- `#define FFLAS_FIELD Givaro::Modular`
- `#define FFLAS_ELT double`
- `#define FFLAS_ELT float`
- `#define FFLAS_ELT int64_t`

## 13.235.1 Macro Definition Documentation

### 13.235.1.1 \_\_FFPACK\_INST\_C

```
#define __FFPACK_INST_C
```

### 13.235.1.2 FFLAS\_COMPILED

```
#define FFLAS_COMPILED
```

### 13.235.1.3 INST\_OR\_DECL

```
#define INST_OR_DECL
```

### 13.235.1.4 FFLAS\_FIELD [1/2]

```
#define FFLAS_FIELD Givaro::ModularBalanced
```

### 13.235.1.5 FFLAS\_ELT [1/6]

```
#define FFLAS_ELT double
```

### 13.235.1.6 FFLAS\_ELT [2/6]

```
#define FFLAS_ELT float
```

### 13.235.1.7 FFLAS\_ELT [3/6]

```
#define FFLAS_ELT int64_t
```

### 13.235.1.8 FFLAS\_FIELD [2/2]

```
#define FFLAS_FIELD Givaro::Modular
```

### 13.235.1.9 FFLAS\_ELT [4/6]

```
#define FFLAS_ELT double
```

### 13.235.1.10 FFLAS\_ELT [5/6]

```
#define FFLAS_ELT float
```

### 13.235.1.11 FFLAS\_ELT [6/6]

```
#define FFLAS_ELT int64_t
```

## 13.236 ffpack\_inst.h File Reference

```
#include "givaro/modular.h"
#include "givaro/modular-balanced.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "ffpack_inst_implem.inl"
```

## Macros

- `#define FFLAS_COMPILED`
- `#define INST_OR_DECL <>`
- `#define FFLAS_FIELD Givaro::ModularBalanced`
- `#define FFLAS_ELT double`
- `#define FFLAS_ELT float`
- `#define FFLAS_ELT int64_t`
- `#define FFLAS_FIELD Givaro::Modular`
- `#define FFLAS_ELT double`
- `#define FFLAS_ELT float`
- `#define FFLAS_ELT int64_t`

## 13.236.1 Macro Definition Documentation

### 13.236.1.1 FFLAS\_COMPILED

```
#define FFLAS_COMPILED
```

### 13.236.1.2 INST\_OR\_DECL

```
#define INST_OR_DECL <>
```

### 13.236.1.3 FFLAS\_FIELD [1/2]

```
#define FFLAS_FIELD Givaro::ModularBalanced
```

### 13.236.1.4 FFLAS\_ELT [1/6]

```
#define FFLAS_ELT double
```

### 13.236.1.5 FFLAS\_ELT [2/6]

```
#define FFLAS_ELT float
```

### 13.236.1.6 FFLAS\_ELT [3/6]

```
#define FFLAS_ELT int64_t
```

### 13.236.1.7 FFLAS\_FIELD [2/2]

```
#define FFLAS_FIELD Givaro::Modular
```

### 13.236.1.8 FFLAS\_ELT [4/6]

```
#define FFLAS_ELT double
```

### 13.236.1.9 FFLAS\_ELT [5/6]

```
#define FFLAS_ELT float
```

### 13.236.1.10 FFLAS\_ELT [6/6]

```
#define FFLAS_ELT int64_t
```



## 13.237 ffpack\_inst\_implem.inl File Reference

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Functions

- void [composePermutationsLLM](#) (size\_t \*MathP, const size\_t \*P1, const size\_t \*P2, const size\_t R, const size\_t N)  
*Computes  $P1 \times \text{Diag}(I_R, P2)$  where  $P1$  is a LAPACK and  $P2$  a LAPACK permutation and store the result in  $\text{MathP}$  as a  $\text{MathPermutation}$  format.*
- void [composePermutationsLLL](#) (size\_t \*P1, const size\_t \*P2, const size\_t R, const size\_t N)  
*Computes  $P1 \times \text{Diag}(I_R, P2)$  where  $P1$  is a LAPACK and  $P2$  a LAPACK permutation and store the result in  $P1$  as a LAPACK permutation.*
- void [composePermutationsMLM](#) (size\_t \*MathP1, const size\_t \*P2, const size\_t R, const size\_t N)  
*Computes  $\text{MathP1} \times \text{Diag}(I_R, P2)$  where  $\text{MathP1}$  is a  $\text{MathPermutation}$  and  $P2$  a LAPACK permutation and store the result in  $\text{MathP1}$  as a  $\text{MathPermutation}$  format.*
- void [cyclic\\_shift\\_mathPerm](#) (size\_t \*P, const size\_t s)
- template<typename Base\_t >  
void [cyclic\\_shift\\_row\\_col](#) (Base\_t \*A, size\_t m, size\_t n, size\_t lda)
- template [INST\\_OR\\_DECL](#) void [cyclic\\_shift\\_row](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, [FFLAS\\_ELT](#) \*A, size\_t m, size\_t n, size\_t lda)
- template [INST\\_OR\\_DECL](#) void [cyclic\\_shift\\_col](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, [FFLAS\\_ELT](#) \*A, size\_t m, size\_t n, size\_t lda)
- template [INST\\_OR\\_DECL](#) void [applyP](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const [FFLAS::FFLAS\\_TRANSPOSE](#) Trans, const size\_t M, const size\_t ibeg, const size\_t iend, [FFLAS\\_ELT](#) \*A, const size\_t lda, const size\_t \*P)
- template [INST\\_OR\\_DECL](#) void [fgetrs](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t R, [FFLAS\\_ELT](#) \*A, const size\_t lda, const size\_t \*P, const size\_t \*Q, [FFLAS\\_ELT](#) \*B, const size\_t ldb, int \*info)
- template [INST\\_OR\\_DECL](#) [FFLAS\\_ELT](#) \* [fgetrs](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t NRHS, const size\_t R, [FFLAS\\_ELT](#) \*A, const size\_t lda, const size\_t \*P, const size\_t \*Q, [FFLAS\\_ELT](#) \*X, const size\_t ldx, const [FFLAS\\_ELT](#) \*B, const size\_t ldb, int \*info)
- template [INST\\_OR\\_DECL](#) size\_t [fgesv](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, [FFLAS\\_ELT](#) \*A, const size\_t lda, [FFLAS\\_ELT](#) \*B, const size\_t ldb, int \*info)
- template [INST\\_OR\\_DECL](#) size\_t [fgesv](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t NRHS, [FFLAS\\_ELT](#) \*A, const size\_t lda, [FFLAS\\_ELT](#) \*X, const size\_t ldx, const [FFLAS\\_ELT](#) \*B, const size\_t ldb, int \*info)
- template [INST\\_OR\\_DECL](#) void [ftrtri](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t N, [FFLAS\\_ELT](#) \*A, const size\_t lda, const size\_t threshold)
- template [INST\\_OR\\_DECL](#) void [trinv\\_left](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const size\_t N, const [FFLAS\\_ELT](#) \*L, const size\_t ldl, [FFLAS\\_ELT](#) \*X, const size\_t ldx)
- template [INST\\_OR\\_DECL](#) void [ftrtrm](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const [FFLAS::FFLAS\\_SIDE](#) side, const [FFLAS::FFLAS\\_DIAG](#) diag, const size\_t N, [FFLAS\\_ELT](#) \*A, const size\_t lda)
- template [INST\\_OR\\_DECL](#) size\_t [PLUQ](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, [FFLAS\\_ELT](#) \*A, const size\_t lda, size\_t \*P, size\_t \*Q)
- template [INST\\_OR\\_DECL](#) size\_t [LUdivine](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const [FFLAS::FFLAS\\_DIAG](#) Diag, const [FFLAS::FFLAS\\_TRANSPOSE](#) trans, const size\_t M, const size\_t N, [FFLAS\\_ELT](#) \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const [FFPACK\\_LU\\_TAG](#) LuTag, const size\_t cutoff)
- template [INST\\_OR\\_DECL](#) size\_t [LUdivine\\_small](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const [FFLAS::FFLAS\\_DIAG](#) Diag, const [FFLAS::FFLAS\\_TRANSPOSE](#) trans, const size\_t M, const size\_t N, [FFLAS\\_ELT](#) \*A, const size\_t lda, size\_t \*P, size\_t \*Q, const [FFPACK\\_LU\\_TAG](#) LuTag)

- template `INST_OR_DECL` `size_t` `LUdivine_gauss` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS::FFLAS_DIAG` Diag, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `size_t` \*P, `size_t` \*Q, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` `size_t` `RowEchelonForm` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `size_t` \*P, `size_t` \*Qt, const bool transform, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` `size_t` `ReducedRowEchelonForm` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `size_t` \*P, `size_t` \*Qt, const bool transform, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` `size_t` `ColumnEchelonForm` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `size_t` \*P, `size_t` \*Qt, const bool transform, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` `size_t` `ReducedColumnEchelonForm` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `size_t` \*P, `size_t` \*Qt, const bool transform, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` `FFLAS_ELT` \* `Invert` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` M, `FFLAS_ELT` \*A, const `size_t` lda, int &nullity)
- template `INST_OR_DECL` `FFLAS_ELT` \* `Invert` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` M, const `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_ELT` \*X, const `size_t` idx, int &nullity)
- template `INST_OR_DECL` `FFLAS_ELT` \* `Invert2` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` M, `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_ELT` \*X, const `size_t` idx, int &nullity)
- template `INST_OR_DECL` `std::list< Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element > & CharPoly` (const `Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > > &R`, `std::list< Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element > &charp`, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_FIELD< FFLAS_ELT >::RandIter &G`, const `FFPACK_CHARPOLY_TAG` CharpTag, const `size_t` degree)
- template `INST_OR_DECL` `Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element & CharPoly` (const `Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > > &R`, `Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element &charp`, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_FIELD< FFLAS_ELT >::RandIter &G`, const `FFPACK_CHARPOLY_TAG` CharpTag, const `size_t` degree)
- template `INST_OR_DECL` `Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element & CharPoly` (const `Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > > &R`, `Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element &charp`, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, const `FFPACK_CHARPOLY_TAG` CharpTag, const `size_t` degree)
- template `INST_OR_DECL` `std::vector< FFLAS_ELT > & MinPoly` (const `FFLAS_FIELD< FFLAS_ELT >` &F, `std::vector< FFLAS_ELT > &minP`, const `size_t` N, const `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_FIELD< FFLAS_ELT >::RandIter &G`)
- template `INST_OR_DECL` `std::vector< FFLAS_ELT > & MinPoly` (const `FFLAS_FIELD< FFLAS_ELT >` &F, `std::vector< FFLAS_ELT > &minP`, const `size_t` N, const `FFLAS_ELT` \*A, const `size_t` lda)
- template `INST_OR_DECL` `std::vector< FFLAS_ELT > & MatVecMinPoly` (const `FFLAS_FIELD< FFLAS_ELT >` &F, `std::vector< FFLAS_ELT > &minP`, const `size_t` N, const `FFLAS_ELT` \*A, const `size_t` lda, const `FFLAS_ELT` \*V, const `size_t` incv)
- template `INST_OR_DECL` `size_t` `KrylovElim` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `size_t` \*P, `size_t` \*Q, const `size_t` deg, `size_t` \*iterates, `size_t` \*inviterates, const `size_t` maxit, `size_t` virt)
- template `INST_OR_DECL` `size_t` `SpecRankProfile` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, const `size_t` deg, `size_t` \*rankProfile)
- template `INST_OR_DECL` `size_t` `Rank` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda)
- template `INST_OR_DECL` bool `IsSingular` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda)
- template `INST_OR_DECL` `FFLAS_ELT` & `Det` (const `FFLAS_FIELD< FFLAS_ELT >` &F, `FFLAS_ELT` &det, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `size_t` \*P, `size_t` \*Q)
- template `INST_OR_DECL` `FFLAS_ELT` & `Det` (const `FFLAS_FIELD< FFLAS_ELT >` &F, `FFLAS_ELT` &det, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, const `FFLAS::ParSeqHelper::Parallel< FFLAS::CuttingStrategy::Recursive, FFLAS::StrategyParameter::Threads >` &parH, `size_t` \*P, `size_t` \*Q)

- template `INST_OR_DECL FFLAS_ELT * Solve` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` M, `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_ELT` \*x, const int incx, const `FFLAS_ELT` \*b, const int incb)
- template `INST_OR_DECL void solveLB` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS::FFLAS_SIDE` Side, const `size_t` M, const `size_t` N, const `size_t` R, `FFLAS_ELT` \*L, const `size_t` ldL, const `size_t` \*Q, `FFLAS_ELT` \*B, const `size_t` ldb)
- template `INST_OR_DECL void solveLB2` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS::FFLAS_SIDE` Side, const `size_t` M, const `size_t` N, const `size_t` R, `FFLAS_ELT` \*L, const `size_t` ldL, const `size_t` \*Q, `FFLAS_ELT` \*B, const `size_t` ldb)
- template `INST_OR_DECL void RandomNullSpaceVector` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS::FFLAS_SIDE` Side, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_ELT` \*X, const `size_t` incX)
- template `INST_OR_DECL size_t NullSpaceBasis` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS::FFLAS_SIDE` Side, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_ELT` \*&NS, `size_t` &ldn, `size_t` &NSdim)
- template `INST_OR_DECL size_t RowRankProfile` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `size_t` \*&rkprofile, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL size_t ColumnRankProfile` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `size_t` \*&rkprofile, const `FFPACK_LU_TAG` LuTag)
- void `RankProfileFromLU` (const `size_t` \*P, const `size_t` N, const `size_t` R, `size_t` \*&rkprofile, const `FFPACK_LU_TAG` LuTag)  
*Recovers the column/row rank profile from the permutation of an LU decomposition.*
- `size_t LeadingSubmatrixRankProfiles` (const `size_t` M, const `size_t` N, const `size_t` R, const `size_t` LSm, const `size_t` LSn, const `size_t` \*P, const `size_t` \*Q, `size_t` \*RRP, `size_t` \*CRP)  
*Recovers the row and column rank profiles of any leading submatrix from the PLUQ decomposition.*
- template `INST_OR_DECL size_t RowRankProfileSubmatrixIndices` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `size_t` \*&rowindices, `size_t` \*&colindices, `size_t` &R)
- template `INST_OR_DECL size_t ColRankProfileSubmatrixIndices` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `size_t` \*&rowindices, `size_t` \*&colindices, `size_t` &R)
- template `INST_OR_DECL size_t RowRankProfileSubmatrix` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_ELT` \*&X, `size_t` &R)
- template `INST_OR_DECL size_t ColRankProfileSubmatrix` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_ELT` \*&X, `size_t` &R)
- template `INST_OR_DECL void getTriangular< FFLAS_FIELD< FFLAS_ELT > >` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS::FFLAS_UPLO` Uplo, const `FFLAS::FFLAS_DIAG` diag, const `size_t` M, const `size_t` N, const `size_t` R, const `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_ELT` \*T, const `size_t` ldT, const bool OnlyNonZeroVectors)
- template `INST_OR_DECL void getTriangular< FFLAS_FIELD< FFLAS_ELT > >` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS::FFLAS_UPLO` Uplo, const `FFLAS::FFLAS_DIAG` diag, const `size_t` M, const `size_t` N, const `size_t` R, `FFLAS_ELT` \*A, const `size_t` lda)
- template `INST_OR_DECL void getEchelonForm< FFLAS_FIELD< FFLAS_ELT > >` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS::FFLAS_UPLO` Uplo, const `FFLAS::FFLAS_DIAG` diag, const `size_t` M, const `size_t` N, const `size_t` R, const `size_t` \*P, const `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_ELT` \*T, const `size_t` ldT, const bool OnlyNonZeroVectors, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL void getEchelonForm< FFLAS_FIELD< FFLAS_ELT > >` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS::FFLAS_UPLO` Uplo, const `FFLAS::FFLAS_DIAG` diag, const `size_t` M, const `size_t` N, const `size_t` R, const `size_t` \*P, `FFLAS_ELT` \*A, const `size_t` lda, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL void getEchelonTransform< FFLAS_FIELD< FFLAS_ELT > >` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS::FFLAS_UPLO` Uplo, const `FFLAS::FFLAS_DIAG` diag, const `size_t` M, const `size_t` N, const `size_t` R, const `size_t` \*P, const `size_t` \*Q, const `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_ELT` \*T, const `size_t` ldT, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL void getReducedEchelonForm< FFLAS_FIELD< FFLAS_ELT > >` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS::FFLAS_UPLO` Uplo, const `size_t` M, const `size_t` N, const `size_t` R, const `size_t` \*P, const `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_ELT` \*T, const `size_t` ldT, const bool OnlyNonZeroVectors, const `FFPACK_LU_TAG` LuTag)

- template `INST_OR_DECL` void `getReducedEchelonForm< FFLAS_FIELD< FFLAS_ELT > >` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS::FFLAS_UPLO` Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, `FFLAS_ELT` \*A, const size\_t lda, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` void `getReducedEchelonTransform< FFLAS_FIELD< FFLAS_ELT > >` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS::FFLAS_UPLO` Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const size\_t \*Q, const `FFLAS_ELT` \*A, const size\_t lda, `FFLAS_ELT` \*T, const size\_t ldt, const `FFPACK_LU_TAG` LuTag)
- void `PLUQtoEchelonPermutation` (const size\_t N, const size\_t R, const size\_t \*P, size\_t \*outPerm)  
*Auxiliary routine: determines the permutation that changes a PLUQ decomposition into a echelon form revealing PLUQ decomposition.*
- template `INST_OR_DECL` `FFLAS_ELT` \* `LQUPtoInverseOfFullRankMinor` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const size\_t rank, `FFLAS_ELT` \*A\_factors, const size\_t lda, const size\_t \*QtPointer, `FFLAS_ELT` \*X, const size\_t ldx)

## 13.238 blockcuts.inl File Reference

```
#include <fflas-ffpack/fflas/fflas_enum.h>
#include <math.h>
#include <cassert>
```

### Data Structures

- struct [Single](#)
- struct [Row](#)
- struct [Column](#)
- struct [Block](#)
- struct [Recursive](#)
- struct [Fixed](#)
- struct [Threads](#)
- struct [Grain](#)
- struct [TwoD](#)
- struct [TwoDAdaptive](#)
- struct [ThreeD](#)
- struct [ThreeDInPlace](#)
- struct [ThreeDAdaptive](#)
- struct [Parallel< C, P >](#)
- struct [Sequential](#)
- struct [Compose< H1, H2 >](#)
- struct [ForStrategy1D< blocksize\\_t, Cut, Param >](#)
- struct [ForStrategy2D< blocksize\\_t, Cut, Param >](#)

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::CuttingStrategy](#)
- namespace [FFLAS::StrategyParameter](#)
- namespace [FFLAS::ParSeqHelper](#)  
*ParSeqHelper for both fgemm and ftrsm.*

### Macros

- #define `__FFLASFFPACK_fflas_blockcuts_INL`
- #define `__FFLASFFPACK_MINBLOCKCUTS` ((size\_t)256)

## Typedefs

- typedef [Row](#) [RNSModulus](#)

## Functions

- template<class Cut = CuttingStrategy::Block, class Strat = StrategyParameter::Threads>  
void [BlockCuts](#) (size\_t &RBLOCKSIZE, size\_t &CBLOCKSIZE, const size\_t m, const size\_t n, const size\_t numthreads)
- template<> void [BlockCuts](#)< [CuttingStrategy::Single](#), [StrategyParameter::Threads](#) > (size\_t &RBLOCKSIZE, size\_t &CBLOCKSIZE, const size\_t m, const size\_t n, const size\_t numthreads)
- template<> void [BlockCuts](#)< [CuttingStrategy::Row](#), [StrategyParameter::Fixed](#) > (size\_t &RBLOCKSIZE, size\_t &CBLOCKSIZE, const size\_t m, const size\_t n, const size\_t numthreads)
- template<> void [BlockCuts](#)< [CuttingStrategy::Row](#), [StrategyParameter::Grain](#) > (size\_t &RBLOCKSIZE, size\_t &CBLOCKSIZE, const size\_t m, const size\_t n, const size\_t grainsize)
- template<> void [BlockCuts](#)< [CuttingStrategy::Block](#), [StrategyParameter::Grain](#) > (size\_t &RBLOCKSIZE, size\_t &CBLOCKSIZE, const size\_t m, const size\_t n, const size\_t grainsize)
- template<> void [BlockCuts](#)< [CuttingStrategy::Column](#), [StrategyParameter::Fixed](#) > (size\_t &RBLOCKSIZE, size\_t &CBLOCKSIZE, const size\_t m, const size\_t n, const size\_t numthreads)
- template<> void [BlockCuts](#)< [CuttingStrategy::Column](#), [StrategyParameter::Grain](#) > (size\_t &RBLOCKSIZE, size\_t &CBLOCKSIZE, const size\_t m, const size\_t n, const size\_t grainsize)
- template<> void [BlockCuts](#)< [CuttingStrategy::Block](#), [StrategyParameter::Fixed](#) > (size\_t &RBLOCKSIZE, size\_t &CBLOCKSIZE, const size\_t m, const size\_t n, const size\_t numthreads)
- template<> void [BlockCuts](#)< [CuttingStrategy::Row](#), [StrategyParameter::Threads](#) > (size\_t &RBLOCKSIZE, size\_t &CBLOCKSIZE, const size\_t m, const size\_t n, const size\_t numthreads)
- template<> void [BlockCuts](#)< [CuttingStrategy::Column](#), [StrategyParameter::Threads](#) > (size\_t &RBLOCKSIZE, size\_t &CBLOCKSIZE, const size\_t m, const size\_t n, const size\_t numthreads)
- template<> void [BlockCuts](#)< [CuttingStrategy::Block](#), [StrategyParameter::Threads](#) > (size\_t &RBLOCKSIZE, size\_t &CBLOCKSIZE, const size\_t m, const size\_t n, const size\_t numthreads)
- template<class Cut = CuttingStrategy::Block, class Param = StrategyParameter::Threads>  
void [BlockCuts](#) (size\_t &rowBlockSize, size\_t &colBlockSize, size\_t &lastRBS, size\_t &lastCBS, size\_t &changeRBS, size\_t &changeCBS, size\_t &numRowBlock, size\_t &numColBlock, size\_t m, size\_t n, const size\_t numthreads)

### 13.238.1 Macro Definition Documentation

#### 13.238.1.1 \_\_FFLASFFPACK\_fflas\_blockcuts\_INL

```
#define __FFLASFFPACK_fflas_blockcuts_INL
```

#### 13.238.1.2 \_\_FFLASFFPACK\_MINBLOCKCUTS

```
#define __FFLASFFPACK_MINBLOCKCUTS ((size_t)256)
```

## 13.239 fflas\_plevel1.h File Reference

```
#include "fflas-ffpack/paladin/parallel.h"
```

## Namespaces

- namespace [FFLAS](#)

## Functions

- template<class [Field](#) >  
void [pfzero](#) (const [Field](#) &F, size\_t m, size\_t n, typename [Field::Element\\_ptr](#) C, size\_t BS=0)

- `template<class Field, class RandIter >`  
`void pfrand (const Field &F, RandIter &G, size_t m, size_t n, typename Field::Element_ptr C, size_t BS=0)`
- `template<class Field, class Cut, class Param >`  
`Field::Element & fdot (const Field &F, const size_t N, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, typename Field::Element &d, const ParSeqHelper::Parallel< Cut, Param > par)`
- `template<typename Field, class Cut, class Param >`  
`Field::Element fdot (const Field &F, const size_t N, typename Field::ConstElement_ptr X, const size_t incX, typename Field::ConstElement_ptr Y, const size_t incY, const ParSeqHelper::Parallel< Cut, Param > par)`

## 13.240 kaapi\_routines.inl File Reference

### Macros

- `#define __FFLASFFPACK_KAAPI_ROUTINES_INL`

### 13.240.1 Macro Definition Documentation

#### 13.240.1.1 \_\_FFLASFFPACK\_KAAPI\_ROUTINES\_INL

```
#define __FFLASFFPACK_KAAPI_ROUTINES_INL
```

## 13.241 parallel.h File Reference

```
#include "fflas-ffpack/config.h"
#include "fflas-ffpack/paladin/blockcuts.inl"
```

### Macros

- `#define __FFLASFFPACK_SEQUENTIAL`
- `#define index_t size_t`
- `#define TASK(M, l)`
- `#define WAIT`
- `#define CHECK_DEPENDENCIES`
- `#define BARRIER`
- `#define PAR_BLOCK`
- `#define SYNCH_GROUP(Args...)`
- `#define THREAD_INDEX 0`
- `#define NUM_THREADS 1`
- `#define SET_THREADS(num_threads)`
- `#define MAX_THREADS 1`
- `#define READ(Args...)`
- `#define WRITE(Args...)`
- `#define READWRITE(Args...)`
- `#define CONSTREFERENCE(...)`
- `#define VALUE(...)`
- `#define BEGIN_PARALLEL_MAIN(Args...)`
- `#define END_PARALLEL_MAIN(void)`
- `#define FORBLOCK1D(iter, m, Helper, Args...)`
- `#define FOR1D(i, m, Helper, Args...)`
- `#define PARFORBLOCK1D(iter, m, Helper, Args...)`
- `#define PARFOR1D(iter, m, Helper, Args...)`
- `#define FORBLOCK2D(iter, m, n, Helper, Args...)`
- `#define FOR2D(i, j, m, n, Helper, Args...)`
- `#define PARFORBLOCK2D(iter, m, n, Helper, Args...)`

- #define [PARFOR2D](#)(i, j, m, n, Helper, Args...)
- #define [COMMA](#) ,
- #define [MODE](#)(...)
- #define [RETURNPARAM](#)(f, P1, Args...)
- #define [NUMARGS](#)(...)
- #define [PP\\_NARG](#)(...)
- #define [PP\\_ARG\\_N](#)( \_1, \_2, \_3, \_4, \_5, \_6, \_7, \_8, \_9, \_10, \_11, \_12, \_13, \_14, \_15, \_16, \_17, \_18, \_19, \_20, \_21, \_22, \_23, \_24, \_25, \_26, \_27, \_28, \_29, \_30, \_31, \_32, \_33, \_34, \_35, \_36, \_37, \_38, \_39, \_40, \_41, \_42, \_43, \_44, \_45, \_46, \_47, \_48, \_49, \_50, \_51, \_52, \_53, \_54, \_55, \_56, \_57, \_58, \_59, \_60, \_61, \_62, \_63, N, ...)
- #define [PP\\_RSEQ\\_N](#)()
- #define [NOSPLIT](#)()
- #define [splitting\\_0](#)()
- #define [splitting\\_1](#)(a)
- #define [splitting\\_2](#)(a, c)
- #define [splitting\\_3](#)(a, b, c)
- #define [splitt](#)(\_1, \_2, \_3, NAME, ...)
- #define [SPLITTER](#)(...)

## 13.241.1 Macro Definition Documentation

### 13.241.1.1 `__FFLASFFPACK_SEQUENTIAL`

```
#define __FFLASFFPACK_SEQUENTIAL
```

### 13.241.1.2 `index_t`

```
#define index_t size_t
```

### 13.241.1.3 `TASK`

```
#define TASK(  
    M,  
    I)
```

**Value:**

```
{I;}
```

### 13.241.1.4 `WAIT`

```
#define WAIT
```

### 13.241.1.5 `CHECK_DEPENDENCIES`

```
#define CHECK_DEPENDENCIES
```

### 13.241.1.6 `BARRIER`

```
#define BARRIER
```

### 13.241.1.7 `PAR_BLOCK`

```
#define PAR_BLOCK
```

### 13.241.1.8 `SYNCH_GROUP`

```
#define SYNCH_GROUP(  
    Args...)
```

**Value:**

```
{{Args}};
```

**13.241.1.9 THREAD\_INDEX**

```
#define THREAD_INDEX 0
```

**13.241.1.10 NUM\_THREADS**

```
#define NUM_THREADS 1
```

**13.241.1.11 SET\_THREADS**

```
#define SET_THREADS(  
    num_threads)
```

**Value:**

```
{ }
```

**13.241.1.12 MAX\_THREADS**

```
#define MAX_THREADS 1
```

**13.241.1.13 READ**

```
#define READ(  
    Args...)
```

**13.241.1.14 WRITE**

```
#define WRITE(  
    Args...)
```

**13.241.1.15 READWRITE**

```
#define READWRITE(  
    Args...)
```

**13.241.1.16 CONSTREFERENCE**

```
#define CONSTREFERENCE(  
    ...)
```

**13.241.1.17 VALUE**

```
#define VALUE(  
    ...)
```

**13.241.1.18 BEGIN\_PARALLEL\_MAIN**

```
#define BEGIN_PARALLEL_MAIN(  
    Args...)
```

**Value:**

```
int main(Args) {
```

**13.241.1.19 END\_PARALLEL\_MAIN**

```
#define END_PARALLEL_MAIN(  
    void)
```

**Value:**

```
return 0; }
```



**13.241.1.20 FORBLOCK1D**

```
#define FORBLOCK1D(
    iter,
    m,
    Helper,
    Args...)

```

**Value:**

```
{ FFLAS::ForStrategy1D<std::remove_const<decltype(m)>::type, typename decltype(Helper)::Cut, typename
    decltype(Helper)::Param> iter(m, Helper); \
    for(iter.initialize(); !iter.isTerminated(); ++iter) \
    {Args; } }

```

**13.241.1.21 FOR1D**

```
#define FOR1D(
    i,
    m,
    Helper,
    Args...)

```

**Value:**

```
FORBLOCK1D(_internal_iterator, m, Helper, \
    for(auto i=_internal_iterator.begin(); i!=_internal_iterator.end(); ++i) \
    { Args; })

```

**13.241.1.22 PARFORBLOCK1D**

```
#define PARFORBLOCK1D(
    iter,
    m,
    Helper,
    Args...)

```

**Value:**

```
for(std::remove_const<decltype(m)>::type iter=0; iter<m; ++iter) \
{ Args; }

```

**13.241.1.23 PARFOR1D**

```
#define PARFOR1D(
    iter,
    m,
    Helper,
    Args...)

```

**Value:**

```
for(std::remove_const<decltype(m)>::type iter=0; iter<m; ++iter) \
{ Args; }

```

**13.241.1.24 FORBLOCK2D**

```
#define FORBLOCK2D(
    iter,
    m,
    n,
    Helper,
    Args...)

```

**Value:**

```
{ FFLAS::ForStrategy2D<std::remove_const<decltype(m)>::type, typename decltype(Helper)::Cut, typename
    decltype(Helper)::Param> iter(m,n,Helper); \
    for(iter.initialize(); !iter.isTerminated(); ++iter) \
    { Args; } }

```

**13.241.1.25 FOR2D**

```
#define FOR2D(
    i,

```

```

        j,
        m,
        n,
        Helper,
        Args...)

```

**Value:**

```

FORBLOCK2D(_internal_iterator, m, n, Helper,
            for(auto i=_internal_iterator.ibegin(); i!=_internal_iterator.iend(); ++i) \
            for(auto j=_internal_iterator.jbegin(); j!=_internal_iterator.jend(); ++j) \
            { Args; })

```

**13.241.1.26 PARFORBLOCK2D**

```

#define PARFORBLOCK2D(
    iter,
    m,
    n,
    Helper,
    Args...)

```

**Value:**

```

FORBLOCK2D(iter, m, n, Helper, Args)

```

**13.241.1.27 PARFOR2D**

```

#define PARFOR2D(
    i,
    j,
    m,
    n,
    Helper,
    Args...)

```

**Value:**

```

FOR2D(i, j, m, n, Helper, Args)

```

**13.241.1.28 COMMA**

```

#define COMMA ,

```

**13.241.1.29 MODE**

```

#define MODE(
    ...)

```

**Value:**

```

__VA_ARGS__

```

**13.241.1.30 RETURNPARAM**

```

#define RETURNPARAM(
    f,
    Pl,
    Args...)

```

**Value:**

```

Pl=f(Args)

```

**13.241.1.31 NUMARGS**

```

#define NUMARGS(
    ...)

```

**Value:**

```

PP_NARG_( __VA_ARGS__, PP_RSEQ_N() )

```

### 13.241.1.32 PP\_NARG\_

```
#define PP_NARG_(  
    ...)
```

**Value:**

[PP\\_ARG\\_N](#)(\_\_VA\_ARGS\_\_)

### 13.241.1.33 PP\_ARG\_N

```
#define PP_ARG_N(  
    _1,  
    _2,  
    _3,  
    _4,  
    _5,  
    _6,  
    _7,  
    _8,  
    _9,  
    _10,  
    _11,  
    _12,  
    _13,  
    _14,  
    _15,  
    _16,  
    _17,  
    _18,  
    _19,  
    _20,  
    _21,  
    _22,  
    _23,  
    _24,  
    _25,  
    _26,  
    _27,  
    _28,  
    _29,  
    _30,  
    _31,  
    _32,  
    _33,  
    _34,  
    _35,  
    _36,  
    _37,  
    _38,  
    _39,  
    _40,  
    _41,  
    _42,  
    _43,  
    _44,  
    _45,  
    _46,  
    _47,  
    _48,  
    _49,
```

```

    _50,
    _51,
    _52,
    _53,
    _54,
    _55,
    _56,
    _57,
    _58,
    _59,
    _60,
    _61,
    _62,
    _63,
    N,
    ...)

```

**Value:**

N

#### 13.241.1.34 PP\_RSEQ\_N

```
#define PP_RSEQ_N()
```

**Value:**

```

63, 62, 61, 60, \
59, 58, 57, 56, 55, 54, 53, 52, 51, 50, \
49, 48, 47, 46, 45, 44, 43, 42, 41, 40, \
39, 38, 37, 36, 35, 34, 33, 32, 31, 30, \
29, 28, 27, 26, 25, 24, 23, 22, 21, 20, \
19, 18, 17, 16, 15, 14, 13, 12, 11, 10, \
9, 8, 7, 6, 5, 4, 3, 2, 1, 0

```

#### 13.241.1.35 NOSPLIT

```
#define NOSPLIT()
```

**Value:**

`FFLAS::ParSeqHelper::Sequential()`

#### 13.241.1.36 splitting\_0

```
#define splitting_0()
```

**Value:**

`FFLAS::ParSeqHelper::Parallel<FFLAS::CuttingStrategy::Block, FFLAS::StrategyParameter::Threads>()`

#### 13.241.1.37 splitting\_1

```
#define splitting_1(
    a)

```

**Value:**

`FFLAS::ParSeqHelper::Parallel<FFLAS::CuttingStrategy::Block, FFLAS::StrategyParameter::Threads>(a)`

#### 13.241.1.38 splitting\_2

```
#define splitting_2(
    a,
    c)

```

**Value:**

`FFLAS::ParSeqHelper::Parallel<FFLAS::CuttingStrategy::Block, c>(a)`

#### 13.241.1.39 splitting\_3

```
#define splitting_3(
    a,

```

*b,*  
*c)*

**Value:**

`FFLAS::ParSeqHelper::Parallel<b,c>(a)`

**13.241.1.40 splitt**

```
#define splitt(  
    _1,  
    _2,  
    _3,  
    NAME,  
    ...)
```

**Value:**

NAME

**13.241.1.41 SPLITTER**

```
#define SPLITTER(  
    ...)
```

**Value:**

`splitt(__VA_ARGS__, splitting_3, splitting_2, splitting_1, splitting_0)(__VA_ARGS__)`

**13.242 pfgemm\_variants.inl File Reference****Namespaces**

- namespace [FFLAS](#)

**Functions**

- `template<class Field, class AlgoT, class FieldTrait >`  
`Field::Element * pfgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb,`  
`const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename`  
`Field::ConstElement_ptr A, const size_t lda, const typename Field::ConstElement_ptr B, const size_t ldb,`  
`const typename Field::Element beta, typename Field::Element *C, const size_t ldc, MMHelper< Field, Al-`  
`goT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Block, StrategyParameter::Threads > > &H)`
- `template<class Field, class AlgoT, class FieldTrait >`  
`Field::Element * pfgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb,`  
`const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename`  
`Field::ConstElement_ptr AA, const size_t lda, const typename Field::ConstElement_ptr BB, const size_t`  
`ldb, const typename Field::Element beta, typename Field::Element *C, const size_t ldc, MMHelper< Field,`  
`AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive, StrategyParameter::ThreeDAdaptive`  
`> > &H)`
- `template<class Field, class AlgoT, class FieldTrait >`  
`Field::Element * pfgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb,`  
`const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename`  
`Field::ConstElement_ptr AA, const size_t lda, const typename Field::ConstElement_ptr BB, const size_t`  
`ldb, const typename Field::Element beta, typename Field::Element *C, const size_t ldc, MMHelper< Field,`  
`AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive, StrategyParameter::TwoDAdaptive >`  
`> &H)`
- `template<class Field, class AlgoT, class FieldTrait >`  
`Field::Element * pfgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb,`  
`const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename`  
`Field::ConstElement_ptr AA, const size_t lda, const typename Field::ConstElement_ptr BB, const size_t`  
`ldb, const typename Field::Element beta, typename Field::Element *C, const size_t ldc, MMHelper< Field,`  
`AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive, StrategyParameter::TwoD > > &H)`
- `template<class Field, class AlgoT, class FieldTrait >`  
`Field::Element_ptr pfgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE`

```
tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename
Field::ConstElement_ptr A, const size_t lda, const typename Field::ConstElement_ptr B, const size_t ldb,
const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field,
AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive, StrategyParameter::ThreeD > > &H)
```

- `template<class Field, class AlgoT, class FieldTrait >`  
`Field::Element * pfgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb,`  
`const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename`  
`Field::ConstElement_ptr A, const size_t lda, const typename Field::ConstElement_ptr B, const size_t ldb,`  
`const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, Al-`  
`goT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive, StrategyParameter::ThreeDInPlace >`  
`> &H)`

## 13.243 pfgemv.inl File Reference

### Namespaces

- namespace `FFLAS`

### Functions

- `template<class Field, class AlgoT, class FieldTrait >`  
`Field::Element_ptr fgemv (const Field &F, const FFLAS_TRANSPOSE ta, const size_t m, const size_t`  
`n, const typename Field::Element alpha, const typename Field::ConstElement_ptr A, const size_t lda,`  
`const typename Field::ConstElement_ptr X, const size_t incX, const typename Field::Element beta, type-`  
`name Field::Element_ptr Y, const size_t incY, MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel<`  
`CuttingStrategy::Recursive, StrategyParameter::Threads > > &H)`
- `template<class Field, class AlgoT, class FieldTrait, class Cut >`  
`Field::Element_ptr fgemv (const Field &F, const FFLAS_TRANSPOSE ta, const size_t m, const size_t`  
`n, const typename Field::Element alpha, const typename Field::ConstElement_ptr A, const size_t lda,`  
`const typename Field::ConstElement_ptr X, const size_t incX, const typename Field::Element beta, type-`  
`name Field::Element_ptr Y, const size_t incY, MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel<`  
`CuttingStrategy::Row, Cut > > &H)`

## 13.244 align-allocator.h File Reference

```
#include "fflas-ffpack/config.h"
```

## 13.245 args-parser.h File Reference

```
#include <fflas-ffpack/fflas-ffpack-config.h>
#include <givaro/givinteger.h>
#include <givaro/givprint.h>
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <cstring>
#include <list>
#include <stdlib.h>
```

### Data Structures

- struct `Argument`

## Namespaces

- namespace [FFLAS](#)

## Macros

- #define [TYPE\\_BOOL](#) [TYPE\\_NONE](#)
- #define [END\\_OF\\_ARGUMENTS](#) { '\0', "\0", "\0", [TYPE\\_NONE](#), NULL }
- #define [type\\_integer](#) long int

## Enumerations

- enum [ArgumentType](#) {  
[TYPE\\_NONE](#) , [TYPE\\_INT](#) , [TYPE\\_UINT64](#) , [TYPE\\_LONGLONG](#) ,  
[TYPE\\_INTEGER](#) , [TYPE\\_DOUBLE](#) , [TYPE\\_INTLIST](#) , [TYPE\\_STR](#) }

## Functions

- void [parseArguments](#) (int argc, char \*\*argv, [Argument](#) \*args, bool printDefaults=true)
- void [printHelpMessage](#) (const char \*program, [Argument](#) \*args, bool printDefaults=false)
- [Argument](#) \* [findArgument](#) ([Argument](#) \*args, char c)
- int [getListArgs](#) (std::list< int > &outlist, std::string &instring)  
*transforms a string list of ints to a list of int string "12,13,15" is turned into list of ints {12,13,15}*
- char \* [getArgumentValue](#) (int argc, char \*\*argv, int i)  
*Get the value of an argument and avoid core dump when no value was given after an argument.*
- std::ostream & [writeCommandString](#) (std::ostream &os, [Argument](#) \*args, const char \*programName=nullptr)  
*writes the values of all arguments, preceded by the programName*

## 13.245.1 Macro Definition Documentation

### 13.245.1.1 TYPE\_BOOL

```
#define TYPE_BOOL TYPE\_NONE
```

### 13.245.1.2 END\_OF\_ARGUMENTS

```
#define END_OF_ARGUMENTS { '\0', "\0", "\0", TYPE\_NONE, NULL }
```

### 13.245.1.3 type\_integer

```
#define type_integer long int
```

## 13.245.2 Enumeration Type Documentation

### 13.245.2.1 ArgumentType

```
enum ArgumentType
```

#### Enumerator

|               |  |
|---------------|--|
| TYPE_NONE     |  |
| TYPE_INT      |  |
| TYPE_UINT64   |  |
| TYPE_LONGLONG |  |
| TYPE_INTEGER  |  |
| TYPE_DOUBLE   |  |
| TYPE_INTLIST  |  |
| TYPE_STR      |  |

### 13.245.3 Function Documentation

#### 13.245.3.1 printHelpMessage()

```
void printHelpMessage (
    const char * program,
    Argument * args,
    bool printDefaults = false)
```

#### 13.245.3.2 findArgument()

```
Argument * findArgument (
    Argument * args,
    char c)
```

#### 13.245.3.3 getListArgs()

```
int getListArgs (
    std::list< int > & outlist,
    std::string & instring)
```

transforms a string list of ints to a list of int string "12,13,15" is turned into list of ints {12,13,15}

##### Parameters

|                 |                      |
|-----------------|----------------------|
| <i>outlist</i>  | list once converted  |
| <i>instring</i> | list to be converted |

##### Returns

status message.

## 13.246 bit\_manipulation.h File Reference

```
#include <givaro/udl.h>
#include "fflas-ffpack/fflas-ffpack-config.h"
```

##### Macros

- #define [\\_\\_has\\_builtin](#)(x)

##### Functions

- int32\_t [clz](#) (uint64\_t val)
- int32\_t [clz](#) (uint32\_t val)
- int32\_t [ctz](#) (uint32\_t val)
- int32\_t [ctz](#) (uint64\_t val)

### 13.246.1 Macro Definition Documentation

#### 13.246.1.1 \_\_has\_builtin

```
#define __has_builtin(
    x)
```

##### Value:

0



## 13.246.2 Function Documentation

### 13.246.2.1 clz() [1/2]

```
int32_t clz (
    uint64_t val) [inline]
```

### 13.246.2.2 clz() [2/2]

```
int32_t clz (
    uint32_t val) [inline]
```

### 13.246.2.3 ctz() [1/2]

```
int32_t ctz (
    uint32_t val) [inline]
```

### 13.246.2.4 ctz() [2/2]

```
int32_t ctz (
    uint64_t val) [inline]
```

## 13.247 cast.h File Reference

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Functions

- template<class T, class CT = const T>  
T [fflas\\_const\\_cast](#) (CT x)

## 13.248 debug.h File Reference

Various utilities for debugging.

```
#include <fflas-ffpack/fflas-ffpack-config.h>
#include <iostream>
#include <sstream>
#include <cmath>
#include <stdexcept>
```

### Data Structures

- class [Failure](#)  
*A precondition failed.*

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [FFLASFFPACK\\_check](#)(check)
- #define [FFLASFFPACK\\_abort](#)(msg)

## Functions

- [Failure](#) & [failure](#) ()
- `template<class T >`  
`bool isOdd (const T &a)`
- `bool isOdd (const float &a)`
- `bool isOdd (const double &a)`

### 13.248.1 Detailed Description

Various utilities for debugging.

**Todo** we should put vector printing elsewhere.

### 13.248.2 Macro Definition Documentation

#### 13.248.2.1 FFLASFFPACK\_check

```
#define FFLASFFPACK_check(  
    check)
```

**Value:**

```
if (!(check)) {\n    FFPACK::failure() (__func__, __FILE__, __LINE__, #check); \n    throw std::runtime_error(#check); \n}
```

#### 13.248.2.2 FFLASFFPACK\_abort

```
#define FFLASFFPACK_abort(  
    msg)
```

**Value:**

```
{\n    FFPACK::failure() (__func__, __FILE__, __LINE__, msg); \n    throw std::runtime_error(msg); \n}
```

## 13.249 fflas\_intrinsic.h File Reference

## 13.250 fflas\_io.h File Reference

```
#include <cstring>\n#include <stdio.h>\n#include <stdlib.h>\n#include <fstream>\n#include "fflas-ffpack/fflas/fflas.h"\n#include "fflas_memory.h"
```

## Namespaces

- namespace [FFLAS](#)

## Enumerations

- enum [FFLAS\\_FORMAT](#) {  
[FflasAuto](#) = 0 , [FflasDense](#) = 1 , [FflasSMS](#) = 2 , [FflasBinary](#) = 3 ,  
[FflasMath](#) = 4 , [FflasMaple](#) = 5 , [FflasSageMath](#) = 6 }

## Functions

- template<class [Field](#) >  
std::ostream & [WriteMatrix](#) (std::ostream &c, const [Field](#) &F, size\_t m, size\_t n, typename [Field::ConstElement\\_ptr](#) A, size\_t lda, [FFLAS\\_FORMAT](#) format, bool column\_major)  
*WriteMatrix: write a matrix to an output stream.*
- void [preamble](#) (std::ifstream &ifs, [FFLAS\\_FORMAT](#) &format)
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [ReadMatrix](#) (std::ifstream &ifs, [Field](#) &F, size\_t &m, size\_t &n, typename [Field::Element\\_ptr](#) &A, [FFLAS\\_FORMAT](#) format=[FflasAuto](#))  
*ReadMatrix: read a matrix from an input stream.*
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [ReadMatrix](#) (const std::string &matrix\_file, [Field](#) &F, size\_t &m, size\_t &n, typename [Field::Element\\_ptr](#) &A, [FFLAS\\_FORMAT](#) format=[FflasAuto](#))  
*ReadMatrix: read a matrix from a file.*
- template<class [Field](#) >  
void [WriteMatrix](#) (std::string &matrix\_file, const [Field](#) &F, int m, int n, typename [Field::ConstElement\\_ptr](#) A, size\_t lda, [FFLAS\\_FORMAT](#) format=[FflasDense](#), bool column\_major=false)  
*WriteMatrix: write a matrix to a file.*
- std::ostream & [WritePermutation](#) (std::ostream &c, const size\_t \*P, size\_t N)  
*WritePermutation: write a permutation matrix to an output stream.*

## 13.251 fflas\_memory.h File Reference

```
#include "fflas-ffpack/utils/align-allocator.h"
#include <givaro/givinteger.h>
```

## Namespaces

- namespace [FFLAS](#)

## Functions

- template<class [Element](#) >  
bool [alignable](#) ()
- template<> bool [alignable](#)< [Givaro::Integer](#) \* > ()
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [fflas\\_new](#) (const [Field](#) &F, const size\_t m, const [Alignment](#) align=[Alignment::DEFAULT](#))
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [fflas\\_new](#) (const [Field](#) &F, const size\_t m, const size\_t n, const [Alignment](#) align=[Alignment::DEFAULT](#))
- template<class [Element](#) >  
[Element](#) \* [fflas\\_new](#) (const size\_t m, const [Alignment](#) align=[Alignment::DEFAULT](#))
- template<class [Element\\_ptr](#) >  
void [fflas\\_delete](#) ([Element\\_ptr](#) A)
- template<class [Ptr](#) , class ... [Args](#)>  
void [fflas\\_delete](#) ([Ptr](#) p, [Args](#) ... args)
- void [prefetch](#) (const int64\_t \*)
- void [getTLBSize](#) (int &tlb)
- void [queryCacheSizes](#) (int &l1, int &l2, int &l3)
- int [queryL1CacheSize](#) ()
- int [queryTopLevelCacheSize](#) ()

## 13.252 fflas\_randommatrix.h File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/utils/debug.h"
#include "fflas-ffpack/fflas/fflas.h"
#include <givaro/givinteger.h>
#include <givaro/givintprime.h>
#include <givaro/givranditer.h>
#include "fflas-ffpack/ffpack/ffpack.h"
```

### Namespaces

- namespace [FFPACK](#)  
*Finite Field PACK Set of elimination based routines for dense linear algebra.*

### Functions

- template<class [Field](#) , class RandIter >  
[Field::Element\\_ptr](#) [NonZeroRandomMatrix](#) (const [Field](#) &F, size\_t m, size\_t n, typename [Field::Element\\_ptr](#) A, size\_t lda, RandIter &G)  
*Random non-zero Matrix.*
- template<class [Field](#) , class RandIter >  
[Field::Element\\_ptr](#) [NonZeroRandomMatrix](#) (const [Field](#) &F, size\_t m, size\_t n, typename [Field::Element\\_ptr](#) A, size\_t lda)  
*Random non-zero Matrix.*
- template<class [Field](#) , class RandIter >  
[Field::Element\\_ptr](#) [RandomMatrix](#) (const [Field](#) &F, size\_t m, size\_t n, typename [Field::Element\\_ptr](#) A, size\_t lda, RandIter &G)  
*Random Matrix.*
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [RandomMatrix](#) (const [Field](#) &F, size\_t m, size\_t n, typename [Field::Element\\_ptr](#) A, size\_t lda)  
*Random Matrix.*
- template<class [Field](#) , class RandIter >  
[Field::Element\\_ptr](#) [RandomTriangularMatrix](#) (const [Field](#) &F, size\_t m, size\_t n, const [FFLAS::FFLAS\\_UPLO](#) UpLo, const [FFLAS::FFLAS\\_DIAG](#) Diag, bool nonsingular, typename [Field::Element\\_ptr](#) A, size\_t lda, RandIter &G)  
*Random Triangular Matrix.*
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [RandomTriangularMatrix](#) (const [Field](#) &F, size\_t m, size\_t n, const [FFLAS::FFLAS\\_UPLO](#) UpLo, const [FFLAS::FFLAS\\_DIAG](#) Diag, bool nonsingular, typename [Field::Element\\_ptr](#) A, size\_t lda)  
*Random Triangular Matrix.*
- size\_t [RandInt](#) (size\_t a, size\_t b)
- template<class [Field](#) , class RandIter >  
[Field::Element\\_ptr](#) [RandomSymmetricMatrix](#) (const [Field](#) &F, size\_t n, bool nonsingular, typename [Field::Element\\_ptr](#) A, size\_t lda, RandIter &G)  
*Random Symmetric Matrix.*
- template<class [Field](#) , class RandIter >  
[Field::Element\\_ptr](#) [RandomMatrixWithRank](#) (const [Field](#) &F, size\_t m, size\_t n, size\_t r, typename [Field::Element\\_ptr](#) A, size\_t lda, RandIter &G)  
*Random Matrix with prescribed rank.*
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [RandomMatrixWithRank](#) (const [Field](#) &F, size\_t m, size\_t n, size\_t r, typename [Field::Element\\_ptr](#) A, size\_t lda)  
*Random Matrix with prescribed rank.*

- `size_t * RandomIndexSubset` (`size_t N`, `size_t R`, `size_t *P`)  
*Pick uniformly at random a sequence of  $R$  distinct elements from the set  $\{0, \dots, N - 1\}$  using Knuth's shuffle.*
- `size_t * RandomPermutation` (`size_t N`, `size_t *P`)  
*Pick uniformly at random a permutation of size  $N$  stored in LAPACK format using Knuth's shuffle.*
- `void RandomRankProfileMatrix` (`size_t M`, `size_t N`, `size_t R`, `size_t *rows`, `size_t *cols`)  
*Pick uniformly at random an  $R$ -subpermutation of dimension  $M \times N$  : a matrix with only  $R$  non-zeros equal to one, in a random rook placement.*
- `void swapval` (`size_t k`, `size_t N`, `size_t *P`, `size_t val`)
- `void RandomSymmetricRankProfileMatrix` (`size_t N`, `size_t R`, `size_t *rows`, `size_t *cols`)  
*Pick uniformly at random a symmetric  $R$ -subpermutation of dimension  $N \times N$  : a symmetric matrix with only  $R$  non-zeros, all equal to one, in a random rook placement.*
- `void RandomLTQSRankProfileMatrix` (`size_t n`, `size_t r`, `size_t t`, `size_t *rows`, `size_t *cols`)
- `template<class Field, class RandIter >`  
`Field::Element_ptr RandomMatrixWithRankandRPM` (`const Field &F`, `size_t M`, `size_t N`, `size_t R`, `typename Field::Element_ptr A`, `size_t lda`, `const size_t *RRP`, `const size_t *CRP`, `RandIter &G`)  
*Random Matrix with prescribed rank and rank profile matrix Creates an  $m \times n$  matrix with random entries and rank  $r$ .*
- `template<class Field >`  
`Field::Element_ptr RandomMatrixWithRankandRPM` (`const Field &F`, `size_t M`, `size_t N`, `size_t R`, `typename Field::Element_ptr A`, `size_t lda`, `const size_t *RRP`, `const size_t *CRP`)  
*Random Matrix with prescribed rank and rank profile matrix Creates an  $m \times n$  matrix with random entries and rank  $r$ .*
- `template<class Field, class RandIter >`  
`Field::Element_ptr RandomSymmetricMatrixWithRankandRPM` (`const Field &F`, `size_t N`, `size_t R`, `typename Field::Element_ptr A`, `size_t lda`, `const size_t *RRP`, `const size_t *CRP`, `RandIter &G`)  
*Random Symmetric Matrix with prescribed rank and rank profile matrix Creates an  $n \times n$  symmetric matrix with random entries and rank  $r$ .*
- `template<class Field >`  
`Field::Element_ptr RandomSymmetricMatrixWithRankandRPM` (`const Field &F`, `size_t M`, `size_t N`, `size_t R`, `typename Field::Element_ptr A`, `size_t lda`, `const size_t *RRP`, `const size_t *CRP`)  
*Random Symmetric Matrix with prescribed rank and rank profile matrix Creates an  $n \times n$  symmetric matrix with random entries and rank  $r$ .*
- `template<class Field, class RandIter >`  
`Field::Element_ptr RandomMatrixWithRankandRandomRPM` (`const Field &F`, `size_t M`, `size_t N`, `size_t R`, `typename Field::Element_ptr A`, `size_t lda`, `RandIter &G`)  
*Random Matrix with prescribed rank, with random rank profile matrix Creates an  $m \times n$  matrix with random entries, rank  $r$  and with a rank profile matrix chosen uniformly at random.*
- `template<class Field >`  
`Field::Element_ptr RandomMatrixWithRankandRandomRPM` (`const Field &F`, `size_t M`, `size_t N`, `size_t R`, `typename Field::Element_ptr A`, `size_t lda`)  
*Random Matrix with prescribed rank, with random rank profile matrix Creates an  $m \times n$  matrix with random entries, rank  $r$  and with a rank profile matrix chosen uniformly at random.*
- `template<class Field, class RandIter >`  
`Field::Element_ptr RandomSymmetricMatrixWithRankandRandomRPM` (`const Field &F`, `size_t N`, `size_t R`, `typename Field::Element_ptr A`, `size_t lda`, `RandIter &G`)  
*Random Symmetric Matrix with prescribed rank, with random rank profile matrix Creates an  $n \times n$  matrix with random entries, rank  $r$  and with a rank profile matrix chosen uniformly at random.*
- `template<class Field >`  
`Field::Element_ptr RandomSymmetricMatrixWithRankandRandomRPM` (`const Field &F`, `size_t N`, `size_t R`, `typename Field::Element_ptr A`, `size_t lda`)  
*Random Symmetric Matrix with prescribed rank, with random rank profile matrix Creates an  $n \times n$  matrix with random entries, rank  $r$  and with a rank profile matrix chosen uniformly at random.*
- `template<class Field >`  
`Field::Element_ptr RandomMatrixWithDet` (`const Field &F`, `size_t n`, `const typename Field::Element d`, `typename Field::Element_ptr A`, `size_t lda`)  
*Random Matrix with prescribed det.*

- `template<class Field , class Randlter >`  
`Field::Element_ptr RandomMatrixWithDet` (const `Field` &F, `size_t` n, const typename `Field::Element` d, type-name `Field::Element_ptr` A, `size_t` lda, `Randlter` &G)  
*Random Matrix with prescribed det.*
- `template<class Field , class Randlter >`  
`Field::Element_ptr RandomLTQSMatrixWithRankandQSorder` (`Field` &F, `size_t` n, `size_t` r, `size_t` t, typename `Field::Element_ptr` A, `size_t` lda, `Randlter` &G)

## 13.253 flimits.h File Reference

```
#include <climits>
#include <limits>
#include <type_traits>
#include <givaro/givinteger.h>
```

### Data Structures

- `struct limits< unsigned char >`
- `struct limits< signed char >`
- `struct limits< char >`
- `struct limits< unsigned short int >`
- `struct limits< short int >`
- `struct limits< unsigned int >`
- `struct limits< int >`
- `struct limits< unsigned long >`
- `struct limits< long >`
- `struct limits< unsigned long long >`
- `struct limits< long long >`
- `struct limits< float >`
- `struct limits< double >`
- `struct limits< Givaro::Integer >`
- `struct limits< Reclnt::ruint< K > >`
- `struct limits< Reclnt::rint< K > >`

### Functions

- `template<class T , class E >`  
`std::enable_if< std::is_signed< T >::value==std::is_signed< E >::value, bool >::type in_range` (E e)
- `template<class T , class E >`  
`std::enable_if<(std::is_signed< T >::value)&&!std::is_signed< E >::value, bool >::type in_range` (E e)
- `template<class T , class E >`  
`std::enable_if<!std::is_signed< T >::value)&&(std::is_signed< E >::value), bool >::type in_range` (E e)

## 13.253.1 Function Documentation

### 13.253.1.1 in\_range() [1/3]

```
template<class T , class E >
std::enable_if< std::is_signed< T >::value==std::is_signed< E >::value, bool >::type in_↵
range (
    E e)
```

**13.253.1.2 in\_range() [2/3]**

```
template<class T , class E >
std::enable_if<(std::is_signed< T >::value)&&!(std::is_signed< E >::value), bool >::type
in_range (
    E e)
```

**13.253.1.3 in\_range() [3/3]**

```
template<class T , class E >
std::enable_if<!(std::is_signed< T >::value)&&(std::is_signed< E >::value), bool >::type
in_range (
    E e)
```

**13.254 Matio.h File Reference**

```
#include <cstring>
#include <stdio.h>
#include <stdlib.h>
#include "fflas_memory.h"
```

**Functions**

- template<class [Field](#) >  
[Field::Element\\_ptr read\\_field](#) (const [Field](#) &F, const char \*mat\_file, size\_t \*tni, size\_t \*tnj)
- template<class [Field](#) >  
std::ostream & [write\\_field](#) (const [Field](#) &F, std::ostream &c, typename [Field::ConstElement\\_ptr](#) E, int n, int m, int id, bool mapleFormat=false, bool column\_major=false)

**13.254.1 Function Documentation****13.254.1.1 read\_field()**

```
template<class Field >
Field::Element\_ptr read\_field (
    const Field & F,
    const char * mat_file,
    size_t * tni,
    size_t * tnj)
```

**13.254.1.2 write\_field()**

```
template<class Field >
std::ostream & write\_field (
    const Field & F,
    std::ostream & c,
    typename Field::ConstElement\_ptr E,
    int n,
    int m,
    int id,
    bool mapleFormat = false,
    bool column_major = false)
```

**13.255 test-utils.h File Reference**

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/utils/debug.h"
```

```
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include <givaro/givinteger.h>
#include <givaro/givintprime.h>
#include <givaro/givranditer.h>
#include <givaro/givtimer.h>
#include <random>
#include <functional>
```

### Namespaces

- namespace [FFLAS](#)
- namespace [FFPACK](#)

*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Functions

- `uint64_t` [getSeed](#) ()
- `template<typename Field >`  
`Field * chooseField (Givaro::Integer q, uint64_t b, uint64_t seed)`
- `template<> Givaro::ZRing< int32_t > * chooseField< Givaro::ZRing< int32_t > > (Givaro::Integer q,`  
`uint64_t b, uint64_t seed)`
- `template<> Givaro::ZRing< int64_t > * chooseField< Givaro::ZRing< int64_t > > (Givaro::Integer q,`  
`uint64_t b, uint64_t seed)`
- `template<> Givaro::ZRing< float > * chooseField< Givaro::ZRing< float > > (Givaro::Integer q, uint64_t`  
`b, uint64_t seed)`
- `template<> Givaro::ZRing< double > * chooseField< Givaro::ZRing< double > > (Givaro::Integer q,`  
`uint64_t b, uint64_t seed)`

## 13.256 timer.h File Reference

```
#include <time.h>
#include <givaro/givtimer.h>
```

### Namespaces

- namespace [FFLAS](#)

### Typedefs

- `typedef Givaro::Timer Timer`
- `typedef Givaro::BaseTimer BaseTimer`
- `typedef Givaro::UserTimer UserTimer`
- `typedef Givaro::SysTimer SysTimer`

## 13.257 cblas.C File Reference

```
#include "fflas-ffpack/config-blas.h"
```

### Macros

- `#define \_\_FFLASFFPACK\_CONFIGURATION`
- `#define \_\_FFLASFFPACK\_HAVE\_CBLAS 1`



**Functions**

- int [main](#) ()

**13.257.1 Macro Definition Documentation****13.257.1.1 \_\_FFLASFFPACK\_CONFIGURATION**

```
#define __FFLASFFPACK_CONFIGURATION
```

**13.257.1.2 \_\_FFLASFFPACK\_HAVE\_CBLAS**

```
#define __FFLASFFPACK_HAVE_CBLAS 1
```

**13.257.2 Function Documentation****13.257.2.1 main()**

```
int main (
    void )
```

**13.258 clapack.C File Reference**

```
#include "fflas-ffpack/config-blas.h"
```

**Macros**

- #define [\\_\\_FFLASFFPACK\\_CONFIGURATION](#)
- #define [\\_\\_FFLASFFPACK\\_HAVE\\_LAPACK](#) 1
- #define [\\_\\_FFLASFFPACK\\_HAVE\\_CLAPACK](#) 1

**Functions**

- int [main](#) ()

**13.258.1 Macro Definition Documentation****13.258.1.1 \_\_FFLASFFPACK\_CONFIGURATION**

```
#define __FFLASFFPACK_CONFIGURATION
```

**13.258.1.2 \_\_FFLASFFPACK\_HAVE\_LAPACK**

```
#define __FFLASFFPACK_HAVE_LAPACK 1
```

**13.258.1.3 \_\_FFLASFFPACK\_HAVE\_CLAPACK**

```
#define __FFLASFFPACK_HAVE_CLAPACK 1
```

**13.258.2 Function Documentation****13.258.2.1 main()**

```
int main (
    void )
```

## 13.259 cuda.C File Reference

```
#include <stdio.h>
#include <cuda_runtime.h>
#include <cusparse.h>
```

### Functions

- int [main](#) ()

### 13.259.1 Function Documentation

#### 13.259.1.1 main()

```
int main (
    void )
```

## 13.260 fblas.C File Reference

```
#include "fflas-ffpack/config-blas.h"
```

### Macros

- #define [\\_\\_FFLASFFPACK\\_CONFIGURATION](#)

### Functions

- void [dgemm\\_](#) (const char \*, const char \*, const int \*, const int \*, const int \*, const double \*, const double \*, const int \*, const double \*, const int \*, const double \*, double \*, const int \*)
- int [main](#) ()

### 13.260.1 Macro Definition Documentation

#### 13.260.1.1 \_\_FFLASFFPACK\_CONFIGURATION

```
#define __FFLASFFPACK_CONFIGURATION
```

### 13.260.2 Function Documentation

#### 13.260.2.1 dgemm\_()

```
void dgemm_ (
    const char * ,
    const char * ,
    const int * ,
    const int * ,
    const int * ,
    const double * ,
    const double * ,
    const int * ,
    const double * ,
    const int * ,
    const double * ,
    double * ,
    const int * )
```

### 13.260.2.2 main()

```
int main (  
    void )
```

## 13.261 lapack.C File Reference

```
#include "fflas-ffpack/config-blas.h"
```

### Macros

- `#define __FFLASFFPACK_CONFIGURATION`
- `#define __FFLASFFPACK_HAVE_LAPACK 1`

### Functions

- `int main ()`

### 13.261.1 Macro Definition Documentation

#### 13.261.1.1 \_\_FFLASFFPACK\_CONFIGURATION

```
#define __FFLASFFPACK_CONFIGURATION
```

#### 13.261.1.2 \_\_FFLASFFPACK\_HAVE\_LAPACK

```
#define __FFLASFFPACK_HAVE_LAPACK 1
```

### 13.261.2 Function Documentation

#### 13.261.2.1 main()

```
int main (  
    void )
```

## 13.262 regression-check.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"  
#include <givaro/modular.h>  
#include "fflas-ffpack/fflas-ffpack.h"
```

### Functions

- `bool check1 ()`
- `bool check2 ()`
- `bool check3 ()`
- `bool check4 ()`
- `bool checkZeroDimCharpoly ()`
- `bool checkZeroDimMinPoly ()`
- `bool gf2ModularBalanced ()`
- `int main ()`

### 13.262.1 Function Documentation

#### 13.262.1.1 check1()

```
bool check1 ()
```

**13.262.1.2 check2()**

```
bool check2 ()
```

**13.262.1.3 check3()**

```
bool check3 ()
```

**13.262.1.4 check4()**

```
bool check4 ()
```

**13.262.1.5 checkZeroDimCharpoly()**

```
bool checkZeroDimCharpoly ()
```

**13.262.1.6 checkZeroDimMinPoly()**

```
bool checkZeroDimMinPoly ()
```

**13.262.1.7 gf2ModularBalanced()**

```
bool gf2ModularBalanced ()
```

**13.262.1.8 main()**

```
int main (
    void )
```

**13.263 test-charpoly-check.C File Reference**

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/fflas_io.h"
```

**Macros**

- `#define` [ENABLE\\_CHECKER\\_charpoly](#) 1
- `#define` [TIME\\_CHECKER\\_CHARPOLY](#) 1

**Functions**

- `template<class Field, class Polynomial >`  
`void printPolynomial (const Field &F, Polynomial &v)`
- `int main (int argc, char **argv)`

**13.263.1 Macro Definition Documentation****13.263.1.1 ENABLE\_CHECKER\_charpoly**

```
#define ENABLE_CHECKER_charpoly 1
```

**13.263.1.2 TIME\_CHECKER\_CHARPOLY**

```
#define TIME_CHECKER_CHARPOLY 1
```

## 13.263.2 Function Documentation

### 13.263.2.1 printPolynomial()

```
template<class Field , class Polynomial >
void printPolynomial (
    const Field & F,
    Polynomial & v)
```

### 13.263.2.2 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.264 test-charpoly.C File Reference

```
#include <iostream>
#include <iomanip>
#include "givaro/modular.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <random>
#include <chrono>
```

### Functions

- template<class Field , class RandIter >  
bool [launch\\_test](#) (const Field &F, size\_t n, typename Field::Element \*A, size\_t lda, size\_t nbit, RandIter &G, FFPACK::FFPACK\_CHARPOLY\_TAG CT)
- template<class Field >  
bool [run\\_with\\_field](#) (const Givaro::Integer p, uint64\_t bits, size\_t n, std::string file, int variant, size\_t iter, uint64\_t seed)
- int [main](#) (int argc, char \*\*argv)

## 13.264.1 Function Documentation

### 13.264.1.1 launch\_test()

```
template<class Field , class RandIter >
bool launch_test (
    const Field & F,
    size_t n,
    typename Field::Element * A,
    size_t lda,
    size_t nbit,
    RandIter & G,
    FFPACK::FFPACK_CHARPOLY_TAG CT)
```

### 13.264.1.2 run\_with\_field()

```
template<class Field >
bool run_with_field (
    const Givaro::Integer p,
    uint64_t bits,
```

```

    size_t n,
    std::string file,
    int variant,
    size_t iter,
    uint64_t seed)

```

### 13.264.1.3 main()

```

int main (
    int argc,
    char ** argv)

```

## 13.265 test-compressQ.C File Reference

```

#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <list>
#include <vector>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"

```

### Typedefs

- typedef Givaro::Modular< double > [Field](#)

### Functions

- template<class T >  
std::ostream & [printvect](#) (std::ostream &o, vector< T > &vect)
- int [main](#) (int argc, char \*\*argv)

## 13.265.1 Typedef Documentation

### 13.265.1.1 Field

```
typedef Givaro::Modular<double> Field
```

## 13.265.2 Function Documentation

### 13.265.2.1 printvect()

```

template<class T >
std::ostream & printvect (
    std::ostream & o,
    vector< T > & vect)

```

[Bug](#) does not belong here

### 13.265.2.2 main()

```

int main (
    int argc,
    char ** argv)

```

## 13.266 test-det-check.C File Reference

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/checkers/checkers_ffpack.h"
#include "fflas-ffpack/checkers/checkers_ffpack.inl"
```

### Macros

- `#define` [ENABLE\\_CHECKER\\_Det](#) 1
- `#define` [TIME\\_CHECKER\\_Det](#) 1

### Functions

- `int` [main](#) (`int` argc, `char **`argv)

### 13.266.1 Macro Definition Documentation

#### 13.266.1.1 [ENABLE\\_CHECKER\\_Det](#)

```
#define ENABLE_CHECKER_Det 1
```

#### 13.266.1.2 [TIME\\_CHECKER\\_Det](#)

```
#define TIME_CHECKER_Det 1
```

### 13.266.2 Function Documentation

#### 13.266.2.1 [main\(\)](#)

```
int main (
    int argc,
    char ** argv)
```

## 13.267 test-det.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iomanip>
#include <iostream>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
```

### Functions

- `template<class` [Field](#) `, class` [RandIter](#) `>`  
  `bool` [test\\_det](#) ([Field](#) &F, `size_t` n, `int` iter, [RandIter](#) &G)
- `int` [main](#) (`int` argc, `char **`argv)

## 13.267.1 Function Documentation

### 13.267.1.1 test\_det()

```
template<class Field , class RandIter >
bool test_det (
    Field & F,
    size_t n,
    int iter,
    RandIter & G)
```

**Todo** test with stride

### 13.267.1.2 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.268 test-echelon.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <iomanip>
#include <givaro/modular-balanced.h>
#include <givaro/udl.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include <random>
#include <chrono>
```

### Macros

- `#define __FFLASFFPACK_SEQUENTIAL`
- `#define __FFLASFFPACK_GAUSSJORDAN_BASECASE 25`
- `#define __FFLASFFPACK_PLUQ_THRESHOLD 25`

### Functions

- `template<class Field , class RandIter >`  
`bool test_colechelon (Field &F, size_t m, size_t n, size_t r, size_t iters, FFPACK::FFPACK_LU_TAG LuTag,`  
`RandIter &G, bool par)`
- `template<class Field , class RandIter >`  
`bool test_rowechelon (Field &F, size_t m, size_t n, size_t r, size_t iters, FFPACK::FFPACK_LU_TAG LuTag,`  
`RandIter &G, bool par)`
- `template<class Field , class RandIter >`  
`bool test_redcolechelon (Field &F, size_t m, size_t n, size_t r, size_t iters, FFPACK::FFPACK_LU_TAG LuTag,`  
`RandIter &G, bool par)`
- `template<class Field , class RandIter >`  
`bool test_redrowechelon (Field &F, size_t m, size_t n, size_t r, size_t iters, FFPACK::FFPACK_LU_TAG LuTag,`  
`RandIter &G, bool par)`
- `template<class Field >`  
`bool run_with_field (Givaro::Integer q, uint64_t b, size_t m, size_t n, size_t r, size_t iters, uint64_t seed)`
- `int main (int argc, char **argv)`



## 13.268.1 Macro Definition Documentation

### 13.268.1.1 \_\_FFLASFFPACK\_SEQUENTIAL

```
#define __FFLASFFPACK_SEQUENTIAL
```

### 13.268.1.2 \_\_FFLASFFPACK\_GAUSSJORDAN\_BASECASE

```
#define __FFLASFFPACK_GAUSSJORDAN_BASECASE 25
```

### 13.268.1.3 \_\_FFLASFFPACK\_PLUQ\_THRESHOLD

```
#define __FFLASFFPACK_PLUQ_THRESHOLD 25
```

## 13.268.2 Function Documentation

### 13.268.2.1 test\_colechelon()

```
template<class Field , class RandIter >
bool test_colechelon (
    Field & F,
    size_t m,
    size_t n,
    size_t r,
    size_t iters,
    FFPACK::FFPACK_LU_TAG LuTag,
    RandIter & G,
    bool par)
```

**Todo** check Ida

### 13.268.2.2 test\_rowechelon()

```
template<class Field , class RandIter >
bool test_rowechelon (
    Field & F,
    size_t m,
    size_t n,
    size_t r,
    size_t iters,
    FFPACK::FFPACK_LU_TAG LuTag,
    RandIter & G,
    bool par)
```

**Todo** check Ida

### 13.268.2.3 test\_redcolechelon()

```
template<class Field , class RandIter >
bool test_redcolechelon (
    Field & F,
    size_t m,
    size_t n,
    size_t r,
    size_t iters,
    FFPACK::FFPACK_LU_TAG LuTag,
    RandIter & G,
    bool par)
```

**Todo** check Ida

### 13.268.2.4 test\_redrowechelon()

```
template<class Field , class RandIter >
bool test_redrowechelon (
    Field & F,
    size_t m,
    size_t n,
    size_t r,
    size_t iters,
    FFPACK::FFPACK_LU_TAG LuTag,
    RandIter & G,
    bool par)
```

**Todo** check Ida

### 13.268.2.5 run\_with\_field()

```
template<class Field >
bool run_with_field (
    Givaro::Integer q,
    uint64_t b,
    size_t m,
    size_t n,
    size_t r,
    size_t iters,
    uint64_t seed)
```

### 13.268.2.6 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.269 test-fadd.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <typeinfo>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "assert.h"
```

### Functions

- template<class Field >  
bool test\_fadd (const Field &F, size\_t m, size\_t k, size\_t n, bool timing, uint64\_t seed)
- template<class Field >  
bool test\_faddin (const Field &F, size\_t m, size\_t k, size\_t n, bool timing, uint64\_t seed)
- template<class Field >  
bool test\_fsub (const Field &F, size\_t m, size\_t k, size\_t n, bool timing, uint64\_t seed)
- template<class Field >  
bool test\_fsubin (const Field &F, size\_t m, size\_t k, size\_t n, bool timing, uint64\_t seed)
- int main (int ac, char \*\*av)

## 13.269.1 Function Documentation

### 13.269.1.1 test\_fadd()

```
template<class Field >
bool test_fadd (
    const Field & F,
    size_t m,
    size_t k,
    size_t n,
    bool timing,
    uint64_t seed)
```

### 13.269.1.2 test\_faddin()

```
template<class Field >
bool test_faddin (
    const Field & F,
    size_t m,
    size_t k,
    size_t n,
    bool timing,
    uint64_t seed)
```

### 13.269.1.3 test\_fsub()

```
template<class Field >
bool test_fsub (
    const Field & F,
    size_t m,
    size_t k,
    size_t n,
    bool timing,
    uint64_t seed)
```

### 13.269.1.4 test\_fsubin()

```
template<class Field >
bool test_fsubin (
    const Field & F,
    size_t m,
    size_t k,
    size_t n,
    bool timing,
    uint64_t seed)
```

### 13.269.1.5 main()

```
int main (
    int ac,
    char ** av)
```

## 13.270 test-fdot.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iomanip>
#include <iostream>
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
```

```
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/paladin/parallel.h"
#include "fflas-ffpack/paladin/fflas_plevel1.h"
#include <givaro/zring.h>
#include <givaro/modular.h>
#include <random>
#include <chrono>
```

## Macros

- #define [ENABLE\\_ALL\\_CHECKINGS](#) 1

## Functions

- template<typename [Field](#) >  
bool [check\\_fdot](#) (const [Field](#) &F, size\_t n, typename [Field::ConstElement\\_ptr](#) a, size\_t inca, typename [Field::ConstElement\\_ptr](#) b, size\_t incb)
- template<class [Field](#) >  
bool [run\\_with\\_field](#) (Givaro::Integer q, size\_t BS, size\_t n, size\_t iters, uint64\_t seed)
- bool [run\\_with\\_Integer](#) (size\_t BS, size\_t n, size\_t iters, uint64\_t seed)
- int [main](#) (int argc, char \*\*argv)

## 13.270.1 Macro Definition Documentation

### 13.270.1.1 [ENABLE\\_ALL\\_CHECKINGS](#)

```
#define ENABLE_ALL_CHECKINGS 1
```

## 13.270.2 Function Documentation

### 13.270.2.1 [check\\_fdot\(\)](#)

```
template<typename Field >
bool check\_fdot (
    const Field & F,
    size_t n,
    typename Field::ConstElement\_ptr a,
    size_t inca,
    typename Field::ConstElement\_ptr b,
    size_t incb)
```

### 13.270.2.2 [run\\_with\\_field\(\)](#)

```
template<class Field >
bool run\_with\_field (
    Givaro::Integer q,
    size_t BS,
    size_t n,
    size_t iters,
    uint64_t seed)
```

### 13.270.2.3 [run\\_with\\_Integer\(\)](#)

```
bool run\_with\_Integer (
    size_t BS,
    size_t n,
    size_t iters,
    uint64_t seed)
```

### 13.270.2.4 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.271 test-fgemm-check.C File Reference

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
```

### Macros

- #define [ENABLE\\_ALL\\_CHECKINGS](#) 1

### Functions

- template<class [Field](#) , class RandIter >  
bool [launch\\_MM\\_dispatch](#) (const [Field](#) &F, const int mm, const int nn, const int kk, const typename [Field::Element](#) alpha, const typename [Field::Element](#) beta, const size\_t iters, RandIter &G)
- template<class [Field](#) >  
bool [run\\_with\\_field](#) (Givaro::Integer q, uint64\_t b, int m, int n, int k, size\_t iters, uint64\_t seed)
- int [main](#) (int argc, char \*\*argv)

### 13.271.1 Macro Definition Documentation

#### 13.271.1.1 ENABLE\_ALL\_CHECKINGS

```
#define ENABLE_ALL_CHECKINGS 1
```

### 13.271.2 Function Documentation

#### 13.271.2.1 launch\_MM\_dispatch()

```
template<class Field , class RandIter >
bool launch_MM_dispatch (
    const Field & F,
    const int mm,
    const int nn,
    const int kk,
    const typename Field::Element alpha,
    const typename Field::Element beta,
    const size_t iters,
    RandIter & G)
```

**Bug** test for ldX equal

**Bug**

**Bug** test for transpo

**Bug**

**Todo** does nbw actually do nbw recursive calls and then call blas (check ?) ?

### 13.271.2.2 run\_with\_field()

```
template<class Field >
bool run_with_field (
    Givaro::Integer q,
    uint64_t b,
    int m,
    int n,
    int k,
    size_t iters,
    uint64_t seed)
```

### 13.271.2.3 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.272 test-fgemm.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include <iomanip>
#include <iostream>
#include <givaro/modular.h>
#include <recint/rint.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <random>
```

### Macros

- #define `ENABLE_CHECKER_fgemm` 1

### Functions

- template<class `Field` >
   
bool `check_MM` (const `Field` &F, const typename `Field::Element_ptr` Cd, enum `FFLAS_TRANSPOSE` &ta, enum `FFLAS_TRANSPOSE` &tb, const size\_t m, const size\_t n, const size\_t k, const typename `Field::Element` &alpha, const typename `Field::Element_ptr` A, size\_t lda, const typename `Field::Element_ptr` B, size\_t ldb, const typename `Field::Element` &beta, const typename `Field::Element_ptr` C, size\_t ldc)
- template<class `Field` , class `RandIter` >
   
bool `launch_MM` (const `Field` &F, const size\_t m, const size\_t n, const size\_t k, const typename `Field::Element` alpha, const typename `Field::Element` beta, const size\_t ldc, const size\_t lda, enum `FFLAS_TRANSPOSE` ta, const size\_t ldb, enum `FFLAS_TRANSPOSE` tb, size\_t iters, int nbw, bool par, `RandIter` &G)
- template<class `Field` , class `RandIter` >
   
bool `launch_MM_dispatch` (const `Field` &F, const int mm, const int nn, const int kk, const typename `Field::Element` alpha, const typename `Field::Element` beta, const size\_t iters, const int nbw, const bool par, `RandIter` &G)
- template<class `Field` >
   
bool `run_with_field` (`Givaro::Integer` q, `uint64_t` b, int m, int n, int k, int nbw, size\_t iters, bool par, size\_t seed)
- int `main` (int argc, char \*\*argv)

## 13.272.1 Macro Definition Documentation

### 13.272.1.1 ENABLE\_CHECKER\_fgemm

```
#define ENABLE_CHECKER_fgemm 1
```

## 13.272.2 Function Documentation

### 13.272.2.1 check\_MM()

```
template<class Field >
bool check_MM (
    const Field & F,
    const typename Field::Element_ptr Cd,
    enum FFLAS_TRANSPOSE & ta,
    enum FFLAS_TRANSPOSE & tb,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename Field::Element & alpha,
    const typename Field::Element_ptr A,
    size_t lda,
    const typename Field::Element_ptr B,
    size_t ldb,
    const typename Field::Element & beta,
    const typename Field::Element_ptr C,
    size_t ldc)
```

### 13.272.2.2 launch\_MM()

```
template<class Field , class RandIter >
bool launch_MM (
    const Field & F,
    const size_t m,
    const size_t n,
    const size_t k,
    const typename Field::Element alpha,
    const typename Field::Element beta,
    const size_t ldc,
    const size_t lda,
    enum FFLAS_TRANSPOSE ta,
    const size_t ldb,
    enum FFLAS_TRANSPOSE tb,
    size_t iters,
    int nbw,
    bool par,
    RandIter & G)
```

### 13.272.2.3 launch\_MM\_dispatch()

```
template<class Field , class RandIter >
bool launch_MM_dispatch (
    const Field & F,
    const int mm,
    const int nn,
    const int kk,
    const typename Field::Element alpha,
    const typename Field::Element beta,
    const size_t iters,
    const int nbw,
```

```
const bool par,
RandIter & G)
```

**Bug** test for ldX equal

**Bug**

**Bug** test for transpo

**Bug**

**Todo** does nbw actually do nbw recursive calls and then call blas (check ?) ?

#### 13.272.2.4 run\_with\_field()

```
template<class Field >
bool run_with_field (
    Givaro::Integer q,
    uint64_t b,
    int m,
    int n,
    int k,
    int nbw,
    size_t iters,
    bool par,
    size_t seed)
```

#### 13.272.2.5 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.273 test-fgemv.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iomanip>
#include <iostream>
#include <givaro/modular.h>
#include <recint/rint.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
```

### Functions

- template<class Field >  
bool **check\_MV** (const Field &F, const typename Field::Element\_ptr Cd, enum FFLAS\_TRANSPOSE &ta, const size\_t m, const size\_t k, const typename Field::Element &alpha, const typename Field::Element\_ptr A, size\_t lda, const typename Field::Element\_ptr X, size\_t incX, const typename Field::Element &beta, const typename Field::Element\_ptr Y, size\_t incY)
- template<class Field, class RandIter >  
bool **launch\_MV** (const Field &F, const size\_t m, const size\_t k, const typename Field::Element &alpha, const typename Field::Element &beta, const size\_t lda, enum FFLAS\_TRANSPOSE ta, const size\_t incX, const size\_t incY, size\_t iters, bool par, RandIter &G)



- `template<class Field , class RandIter >`  
`bool launch_MV_dispatch` (const `Field` &`F`, const int `mm`, const int `kk`, const typename `Field::Element` `alpha`, const typename `Field::Element` `beta`, const `size_t` `iters`, const bool `par`, `RandIter` &`G`)
- `template<class Field >`  
`bool run_with_field` (`Givaro::Integer` `q`, `uint64_t` `b`, int `m`, int `k`, `size_t` `iters`, bool `par`, `uint64_t` `seed`)
- int `main` (int `argc`, char \*\*`argv`)

## 13.273.1 Function Documentation

### 13.273.1.1 check\_MV()

```
template<class Field >
bool check_MV (
    const Field & F,
    const typename Field::Element_ptr Cd,
    enum FFLAS_TRANSPOSE & ta,
    const size_t m,
    const size_t k,
    const typename Field::Element & alpha,
    const typename Field::Element_ptr A,
    size_t lda,
    const typename Field::Element_ptr X,
    size_t incX,
    const typename Field::Element & beta,
    const typename Field::Element_ptr Y,
    size_t incY)
```

### 13.273.1.2 launch\_MV()

```
template<class Field , class RandIter >
bool launch_MV (
    const Field & F,
    const size_t m,
    const size_t k,
    const typename Field::Element alpha,
    const typename Field::Element beta,
    const size_t lda,
    enum FFLAS_TRANSPOSE ta,
    const size_t incX,
    const size_t incY,
    size_t iters,
    bool par,
    RandIter & G)
```

### 13.273.1.3 launch\_MV\_dispatch()

```
template<class Field , class RandIter >
bool launch_MV_dispatch (
    const Field & F,
    const int mm,
    const int kk,
    const typename Field::Element alpha,
    const typename Field::Element beta,
    const size_t iters,
    const bool par,
    RandIter & G)
```

### 13.273.1.4 run\_with\_field()

```
template<class Field >
bool run_with_field (
    Givaro::Integer q,
    uint64_t b,
    int m,
    int k,
    size_t iters,
    bool par,
    uint64_t seed)
```

### 13.273.1.5 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.274 test-fger.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iomanip>
#include <iostream>
#include <givaro/modular-integral.h>
#include <givaro/modular-balanced.h>
#include <givaro/givintprime.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/fflas_io.h"
```

### Macros

- #define `TIME` 1

### Functions

- template<class `Field` >  
bool `check_fger` (const `Field` &F, const typename `Field::Element_ptr` Cd, const size\_t m, const size\_t n, const typename `Field::Element` &alpha, const typename `Field::Element_ptr` x, const size\_t incx, const typename `Field::Element_ptr` y, const size\_t incy, const typename `Field::Element_ptr` C, const size\_t ldc)
- template<class `Field` , class `RandIter` >  
bool `launch_fger` (const `Field` &F, const size\_t m, const size\_t n, const typename `Field::Element` alpha, const size\_t ldc, const size\_t inca, const size\_t incb, size\_t iters, `RandIter` &G)
- template<class `Field` , class `RandIter` >  
bool `launch_fger_dispatch` (const `Field` &F, const size\_t nn, const typename `Field::Element` alpha, const size\_t iters, `RandIter` &G)
- template<class `Field` >  
bool `run_with_field` (int64\_t q, uint64\_t b, size\_t n, size\_t iters, uint64\_t seed)
- int `main` (int argc, char \*\*argv)

## 13.274.1 Macro Definition Documentation

### 13.274.1.1 TIME

```
#define TIME 1
```

## 13.274.2 Function Documentation

### 13.274.2.1 check\_fger()

```
template<class Field >
bool check_fger (
    const Field & F,
    const typename Field::Element_ptr Cd,
    const size_t m,
    const size_t n,
    const typename Field::Element & alpha,
    const typename Field::Element_ptr x,
    const size_t incx,
    const typename Field::Element_ptr y,
    const size_t incy,
    const typename Field::Element_ptr C,
    const size_t ldc)
```

### 13.274.2.2 launch\_fger()

```
template<class Field , class RandIter >
bool launch_fger (
    const Field & F,
    const size_t m,
    const size_t n,
    const typename Field::Element alpha,
    const size_t ldc,
    const size_t inca,
    const size_t incb,
    size_t iters,
    RandIter & G)
```

### 13.274.2.3 launch\_fger\_dispatch()

```
template<class Field , class RandIter >
bool launch_fger_dispatch (
    const Field & F,
    const size_t nn,
    const typename Field::Element alpha,
    const size_t iters,
    RandIter & G)
```

**Bug** test for incx equal

**Bug**

**Bug** test for transpo

**Bug**

**Todo** does nbw actually do nbw recursive calls and then call blas (check ?) ?

### 13.274.2.4 run\_with\_field()

```
template<class Field >
bool run_with_field (
    int64_t q,
    uint64_t b,
    size_t n,
    size_t iters,
    uint64_t seed)
```

### 13.274.2.5 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.275 test-fgesv.C File Reference

```
#include <iostream>
#include <iomanip>
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>
```

### Functions

- template<class [Field](#) , class RandIter >  
bool [test\\_square\\_fgesv](#) ([Field](#) &F, [FFLAS\\_SIDE](#) side, string fileA, string fileB, size\_t m, size\_t k, size\_t r, RandIter &G)
- template<class [Field](#) , class RandIter >  
bool [test\\_rect\\_fgesv](#) ([Field](#) &F, [FFLAS\\_SIDE](#) side, string fileA, string fileB, size\_t m, size\_t n, size\_t k, size\_t r, RandIter &G)
- template<class [Field](#) >  
bool [run\\_with\\_field](#) (Givaro::Integer q, uint64\_t b, size\_t m, size\_t n, size\_t k, size\_t r, size\_t iters, string fileA, string fileB, uint64\_t &seed)
- int [main](#) (int argc, char \*\*argv)

### 13.275.1 Function Documentation

#### 13.275.1.1 test\_square\_fgesv()

```
template<class Field , class RandIter >
bool test_square_fgesv (
    Field & F,
    FFLAS\_SIDE side,
    string fileA,
    string fileB,
    size_t m,
    size_t k,
    size_t r,
    RandIter & G)
```

#### 13.275.1.2 test\_rect\_fgesv()

```
template<class Field , class RandIter >
bool test_rect_fgesv (
    Field & F,
    FFLAS\_SIDE side,
    string fileA,
    string fileB,
    size_t m,
    size_t n,
    size_t k,
    size_t r,
    RandIter & G)
```

### 13.275.1.3 run\_with\_field()

```
template<class Field >
bool run_with_field (
    Givaro::Integer q,
    uint64_t b,
    size_t m,
    size_t n,
    size_t k,
    size_t r,
    size_t iters,
    string fileA,
    string fileB,
    uint64_t & seed)
```

### 13.275.1.4 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.276 test-finit.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <typeinfo>
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "assert.h"
#include <random>
#include <chrono>
```

### Functions

- template<class Field >  
bool [test\\_freduce](#) (const Field &F, size\_t m, size\_t k, size\_t n, bool timing, uint64\_t seed)
- template<class Field >  
bool [run\\_with\\_field](#) (Givaro::Integer q, size\_t b, size\_t m, size\_t k, size\_t n, size\_t iters, bool timing, uint64\_t seed)
- int [main](#) (int ac, char \*\*av)

## 13.276.1 Function Documentation

### 13.276.1.1 test\_freduce()

```
template<class Field >
bool test_freduce (
    const Field & F,
    size_t m,
    size_t k,
    size_t n,
    bool timing,
    uint64_t seed)
```

### 13.276.1.2 run\_with\_field()

```
template<class Field >
bool run_with_field (
    Givaro::Integer q,
    size_t b,
    size_t m,
    size_t k,
    size_t n,
    size_t iters,
    bool timing,
    uint64_t seed)
```

### 13.276.1.3 main()

```
int main (
    int ac,
    char ** av)
```

## 13.277 test-fscal.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <typeinfo>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "assert.h"
```

### Functions

- template<class Field, class RandIter >  
bool test\_fscal (const Field &F, const typename Field::Element &alpha, size\_t m, size\_t k, size\_t n, bool timing, RandIter &G)
- template<class Field >  
bool test\_fscal (const Field &F, size\_t m, size\_t k, size\_t n, bool timing, uint64\_t seed)
- template<class Field, class RandIter >  
bool test\_fscal\_in (const Field &F, const typename Field::Element &alpha, size\_t m, size\_t k, size\_t n, bool timing, RandIter &G)
- template<class Field >  
bool test\_fscal\_in (const Field &F, size\_t m, size\_t k, size\_t n, bool timing, uint64\_t seed)
- int main (int ac, char \*\*av)

### 13.277.1 Function Documentation

#### 13.277.1.1 test\_fscal() [1/2]

```
template<class Field, class RandIter >
bool test_fscal (
    const Field & F,
    const typename Field::Element & alpha,
    size_t m,
    size_t k,
    size_t n,
    bool timing,
    RandIter & G)
```

**13.277.1.2 test\_fscal() [2/2]**

```
template<class Field >
bool test_fscal (
    const Field & F,
    size_t m,
    size_t k,
    size_t n,
    bool timing,
    uint64_t seed)
```

**13.277.1.3 test\_fscalin() [1/2]**

```
template<class Field , class RandIter >
bool test_fscalin (
    const Field & F,
    const typename Field::Element & alpha,
    size_t m,
    size_t k,
    size_t n,
    bool timing,
    RandIter & G)
```

**13.277.1.4 test\_fscalin() [2/2]**

```
template<class Field >
bool test_fscalin (
    const Field & F,
    size_t m,
    size_t k,
    size_t n,
    bool timing,
    uint64_t seed)
```

**13.277.1.5 main()**

```
int main (
    int ac,
    char ** av)
```

**13.278 test-fsyr2k.C File Reference**

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iomanip>
#include <iostream>
#include <random>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>
```

**Macros**

- #define `ENABLE_ALL_CHECKINGS` 1

## Functions

- `template<typename Field , class RandIter >`  
`bool check\_fsyr2k (const Field &F, size_t n, size_t k, const typename Field::Element &alpha, const typename Field::Element &beta, FFLAS::FFLAS\_UPLO uplo, FFLAS::FFLAS\_TRANSPOSE trans, RandIter &Rand)`
- `template<class Field >`  
`bool run\_with\_field (Givaro::Integer q, size_t b, size_t n, size_t k, int a, int c, size_t iters, uint64_t seed)`
- `int main (int argc, char **argv)`

## 13.278.1 Macro Definition Documentation

### 13.278.1.1 `ENABLE_ALL_CHECKINGS`

```
#define ENABLE_ALL_CHECKINGS 1
```

## 13.278.2 Function Documentation

### 13.278.2.1 `check_fsyr2k()`

```
template<typename Field , class RandIter >
bool check\_fsyr2k (
    const Field & F,
    size_t n,
    size_t k,
    const typename Field::Element & alpha,
    const typename Field::Element & beta,
    FFLAS::FFLAS\_UPLO uplo,
    FFLAS::FFLAS\_TRANSPOSE trans,
    RandIter & Rand)
```

### 13.278.2.2 `run_with_field()`

```
template<class Field >
bool run\_with\_field (
    Givaro::Integer q,
    size_t b,
    size_t n,
    size_t k,
    int a,
    int c,
    size_t iters,
    uint64_t seed)
```

### 13.278.2.3 `main()`

```
int main (
    int argc,
    char ** argv)
```

## 13.279 test-fsyrk.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iomanip>
#include <iostream>
#include <random>
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
```



```
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>
```

## Macros

- #define [ENABLE\\_ALL\\_CHECKINGS](#) 1

## Functions

- template<typename [Field](#) , class RandIter >  
bool [check\\_fsyrc](#) (const [Field](#) &F, size\_t n, size\_t k, size\_t w, const typename [Field::Element](#) &alpha, const typename [Field::Element](#) &beta, [FFLAS::FFLAS\\_UPLO](#) uplo, [FFLAS::FFLAS\\_TRANSPOSE](#) trans, RandIter &Rand)
- template<typename [Field](#) , class RandIter >  
bool [check\\_fsyrc\\_diag](#) (const [Field](#) &F, size\_t n, size\_t k, const typename [Field::Element](#) &alpha, const typename [Field::Element](#) &beta, [FFLAS::FFLAS\\_UPLO](#) uplo, [FFLAS::FFLAS\\_TRANSPOSE](#) trans, RandIter &Rand)
- template<typename [Field](#) , class RandIter >  
bool [check\\_fsyrc\\_bkdiag](#) (const [Field](#) &F, size\_t n, size\_t k, const typename [Field::Element](#) &alpha, const typename [Field::Element](#) &beta, [FFLAS\\_UPLO](#) uplo, [FFLAS\\_TRANSPOSE](#) trans, RandIter &Rand)
- template<class [Field](#) , class RandIter >  
bool [check\\_computeS1S2](#) (const [Field](#) &F, size\_t N, size\_t K, [FFLAS\\_TRANSPOSE](#) trans, RandIter &G)
- template<class [Field](#) >  
bool [run\\_with\\_field](#) (Givaro::Integer q, size\_t b, size\_t n, size\_t k, size\_t w, int a, int c, size\_t iters, uint64\_t seed)
- int [main](#) (int argc, char \*\*argv)

## 13.279.1 Macro Definition Documentation

### 13.279.1.1 [ENABLE\\_ALL\\_CHECKINGS](#)

```
#define ENABLE\_ALL\_CHECKINGS 1
```

## 13.279.2 Function Documentation

### 13.279.2.1 [check\\_fsyrc\(\)](#)

```
template<typename Field , class RandIter >
bool check\_fsyrc (
    const Field & F,
    size_t n,
    size_t k,
    size_t w,
    const typename Field::Element & alpha,
    const typename Field::Element & beta,
    FFLAS::FFLAS\_UPLO uplo,
    FFLAS::FFLAS\_TRANSPOSE trans,
    RandIter & Rand)
```

### 13.279.2.2 [check\\_fsyrc\\_diag\(\)](#)

```
template<typename Field , class RandIter >
bool check\_fsyrc\_diag (
    const Field & F,
    size_t n,
    size_t k,
    const typename Field::Element & alpha,
    const typename Field::Element & beta,
```

```

FFLAS::FFLAS_UPLO uplo,
FFLAS::FFLAS_TRANSPOSE trans,
RandIter & Rand)

```

### 13.279.2.3 check\_fsyrk\_bkdiag()

```

template<typename Field , class RandIter >
bool check_fsyrk_bkdiag (
    const Field & F,
    size_t n,
    size_t k,
    const typename Field::Element & alpha,
    const typename Field::Element & beta,
    FFLAS_UPLO uplo,
    FFLAS_TRANSPOSE trans,
    RandIter & Rand)

```

### 13.279.2.4 check\_computeS1S2()

```

template<class Field , class RandIter >
bool check_computeS1S2 (
    const Field & F,
    size_t N,
    size_t K,
    FFLAS_TRANSPOSE trans,
    RandIter & G)

```

### 13.279.2.5 run\_with\_field()

```

template<class Field >
bool run_with_field (
    Givaro::Integer q,
    size_t b,
    size_t n,
    size_t k,
    size_t w,
    int a,
    int c,
    size_t iters,
    uint64_t seed)

```

### 13.279.2.6 main()

```

int main (
    int argc,
    char ** argv)

```

## 13.280 test-fsytrf.C File Reference

```

#include <iostream>
#include <iterator>
#include <vector>
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utlis/args-parser.h"
#include <iomanip>
#include <random>
#include <chrono>

```

```
#include <givaro/modular.h>
#include "fflas-ffpack/utils/test-utils.h"
```

## Functions

- `template<typename T >`  
`std::ostream & operator<< (std::ostream &os, const std::vector< T > &x)`
- `template<class Field , class RandIter >`  
`bool test_RPM_fsytrf (Field &F, FFLAS_UPLO uplo, string file, size_t n, size_t r, RandIter &G, size_t threshold)`
- `template<class Field , class RandIter >`  
`bool test_generic_fsytrf (Field &F, FFLAS_UPLO uplo, string file, size_t n, RandIter &G, size_t threshold)`
- `template<class Field >`  
`bool run_with_field (Givaro::Integer q, uint64_t b, size_t n, size_t r, size_t iters, string file, size_t threshold, uint64_t &seed)`
- `int main (int argc, char **argv)`

## 13.280.1 Function Documentation

### 13.280.1.1 operator<<()

```
template<typename T >
std::ostream & operator<< (
    std::ostream & os,
    const std::vector< T > & x)
```

### 13.280.1.2 test\_RPM\_fsytrf()

```
template<class Field , class RandIter >
bool test_RPM_fsytrf (
    Field & F,
    FFLAS_UPLO uplo,
    string file,
    size_t n,
    size_t r,
    RandIter & G,
    size_t threshold)
```

### 13.280.1.3 test\_generic\_fsytrf()

```
template<class Field , class RandIter >
bool test_generic_fsytrf (
    Field & F,
    FFLAS_UPLO uplo,
    string file,
    size_t n,
    RandIter & G,
    size_t threshold)
```

### 13.280.1.4 run\_with\_field()

```
template<class Field >
bool run_with_field (
    Givaro::Integer q,
    uint64_t b,
    size_t n,
    size_t r,
    size_t iters,
```

```

    string file,
    size_t threshold,
    uint64_t & seed)

```

### 13.280.1.5 main()

```

int main (
    int argc,
    char ** argv)

```

## 13.281 test-fftrmm.C File Reference

```

#include "fflas-ffpack/fflas-ffpack-config.h"
#include <givaro/modular-integer.h>
#include <iomanip>
#include <iostream>
#include <random>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>

```

### Macros

- `#define \_\_FFLASFFPACK\_SEQUENTIAL`

### Functions

- `template<typename Field , class RandIter >`  
`bool check\_fftrmm (const Field &F, size_t m, size_t n, const typename Field::Element &alpha,`  
`FFLAS::FFLAS\_SIDE side, FFLAS::FFLAS\_UPLO uplo, FFLAS::FFLAS\_TRANSPOSE trans, FFLAS::FFLAS\_DIAG`  
`diag, RandIter &Rand)`
- `template<class Field >`  
`bool run\_with\_field (Givaro::Integer q, size_t b, size_t m, size_t n, uint64_t a, size_t iters, uint64_t seed)`
- `int main (int argc, char **argv)`

## 13.281.1 Macro Definition Documentation

### 13.281.1.1 [\\_\\_FFLASFFPACK\\_SEQUENTIAL](#)

```
#define \_\_FFLASFFPACK\_SEQUENTIAL
```

## 13.281.2 Function Documentation

### 13.281.2.1 [check\\_fftrmm\(\)](#)

```

template<typename Field , class RandIter >
bool check\_fftrmm (
    const Field & F,
    size_t m,
    size_t n,
    const typename Field::Element & alpha,
    FFLAS::FFLAS\_SIDE side,
    FFLAS::FFLAS\_UPLO uplo,
    FFLAS::FFLAS\_TRANSPOSE trans,

```

```

    FFLAS::FFLAS_DIAG diag,
    RandIter & Rand)

```

### 13.281.2.2 run\_with\_field()

```

template<class Field >
bool run_with_field (
    Givaro::Integer q,
    size_t b,
    size_t m,
    size_t n,
    uint64_t a,
    size_t iters,
    uint64_t seed)

```

### 13.281.2.3 main()

```

int main (
    int argc,
    char ** argv)

```

## 13.282 test-ffrmv.C File Reference

```

#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iomanip>
#include <iostream>
#include <chrono>
#include <random>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>

```

### Macros

- #define [\\_\\_FFLASFFPACK\\_SEQUENTIAL](#)
- #define [ENABLE\\_ALL\\_CHECKINGS](#) 1

### Functions

- template<typename [Field](#) , class RandIter >  
bool [check\\_ffrmv](#) (const [Field](#) &F, size\_t n, [FFLAS\\_UPLO](#) uplo, [FFLAS\\_TRANSPOSE](#) trans, [FFLAS\\_DIAG](#) diag, RandIter &Rand)
- template<class [Field](#) >  
bool [run\\_with\\_field](#) (Givaro::Integer q, size\_t b, size\_t n, size\_t iters, uint64\_t seed)
- int [main](#) (int argc, char \*\*argv)

## 13.282.1 Macro Definition Documentation

### 13.282.1.1 \_\_FFLASFFPACK\_SEQUENTIAL

```

#define __FFLASFFPACK_SEQUENTIAL

```

### 13.282.1.2 ENABLE\_ALL\_CHECKINGS

```

#define ENABLE_ALL_CHECKINGS 1

```

## 13.282.2 Function Documentation

### 13.282.2.1 check\_ftrmv()

```
template<typename Field , class RandIter >
bool check_ftrmv (
    const Field & F,
    size_t n,
    FFLAS_UPLO uplo,
    FFLAS_TRANSPOSE trans,
    FFLAS_DIAG diag,
    RandIter & Rand)
```

### 13.282.2.2 run\_with\_field()

```
template<class Field >
bool run_with_field (
    Givaro::Integer q,
    size_t b,
    size_t n,
    size_t iters,
    uint64_t seed)
```

### 13.282.2.3 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.283 test-ftrsm-check.C File Reference

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/fflas_io.h"
```

### Macros

- #define [ENABLE\\_ALL\\_CHECKINGS](#) 1

### Functions

- int [main](#) (int argc, char \*\*argv)

## 13.283.1 Macro Definition Documentation

### 13.283.1.1 ENABLE\_ALL\_CHECKINGS

```
#define ENABLE_ALL_CHECKINGS 1
```

## 13.283.2 Function Documentation

### 13.283.2.1 main()

```
int main (
    int argc,
```

```
char ** argv)
```

## 13.284 test-ftrsm.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <givaro/modular-integer.h>
#include <iomanip>
#include <iostream>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
#include <random>
```

### Macros

- `#define __FFLASFFPACK_SEQUENTIAL`
- `#define ENABLE_ALL_CHECKINGS 1`

### Functions

- `template<typename Field, class RandIter >`  
`bool check_ftrsm (const Field &F, size_t m, size_t n, const typename Field::Element &alpha,`  
`FFLAS::FFLAS_SIDE side, FFLAS::FFLAS_UPLO uplo, FFLAS::FFLAS_TRANSPOSE trans, FFLAS::FFLAS_DIAG`  
`diag, RandIter &Rand)`
- `template<class Field >`  
`bool run_with_field (Givaro::Integer q, size_t b, size_t m, size_t n, uint64_t a, size_t iters, uint64_t seed)`
- `int main (int argc, char **argv)`

## 13.284.1 Macro Definition Documentation

### 13.284.1.1 \_\_FFLASFFPACK\_SEQUENTIAL

```
#define __FFLASFFPACK_SEQUENTIAL
```

### 13.284.1.2 ENABLE\_ALL\_CHECKINGS

```
#define ENABLE_ALL_CHECKINGS 1
```

## 13.284.2 Function Documentation

### 13.284.2.1 check\_ftrsm()

```
template<typename Field, class RandIter >
bool check_ftrsm (
    const Field & F,
    size_t m,
    size_t n,
    const typename Field::Element & alpha,
    FFLAS::FFLAS_SIDE side,
    FFLAS::FFLAS_UPLO uplo,
    FFLAS::FFLAS_TRANSPOSE trans,
    FFLAS::FFLAS_DIAG diag,
    RandIter & Rand)
```

### 13.284.2.2 run\_with\_field()

```
template<class Field >
bool run_with_field (
    Givaro::Integer q,
    size_t b,
    size_t m,
    size_t n,
    uint64_t a,
    size_t iters,
    uint64_t seed)
```

### 13.284.2.3 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.285 test-ftsyr2k.C File Reference

```
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <givaro/modular-integer.h>
#include <iomanip>
#include <iostream>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
#include <random>
```

### Macros

- `#define ENABLE_ALL_CHECKINGS 1`

### Functions

- `template<typename Field , class RandIter >`  
`bool check_ftsyr2k (const Field &F, size_t n, FFLAS::FFLAS_UPLO uplo, FFLAS::FFLAS_DIAG diagA,`  
`RandIter &Rand)`
- `template<class Field >`  
`bool run_with_field (Givaro::Integer q, size_t b, size_t n, size_t iters, uint64_t seed)`
- `int main (int argc, char **argv)`

## 13.285.1 Macro Definition Documentation

### 13.285.1.1 ENABLE\_ALL\_CHECKINGS

```
#define ENABLE_ALL_CHECKINGS 1
```

## 13.285.2 Function Documentation

### 13.285.2.1 check\_ftsyr2k()

```
template<typename Field , class RandIter >
bool check_ftsyr2k (
```



```

    const Field & F,
    size_t n,
    FFLAS::FFLAS_UPLO uplo,
    FFLAS::FFLAS_DIAG diagA,
    RandIter & Rand)

```

### 13.285.2.2 run\_with\_field()

```

template<class Field >
bool run_with_field (
    Givaro::Integer q,
    size_t b,
    size_t n,
    size_t iters,
    uint64_t seed)

```

### 13.285.2.3 main()

```

int main (
    int argc,
    char ** argv)

```

## 13.286 test-ffrstr.C File Reference

```

#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <givaro/modular-integer.h>
#include <iomanip>
#include <iostream>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
#include <random>

```

### Macros

- #define `ENABLE_ALL_CHECKINGS` 1

### Functions

- template<typename `Field` , class `RandIter` >  
 bool `check_ffrstr` (const `Field` &F, size\_t n, `FFLAS::FFLAS_SIDE` side, `FFLAS::FFLAS_UPLO` uplo, `FFLAS::FFLAS_DIAG` diagA, `FFLAS::FFLAS_DIAG` diagB, `RandIter` &Rand)
- template<class `Field` >  
 bool `run_with_field` (`Givaro::Integer` q, size\_t b, size\_t n, size\_t iters, uint64\_t seed)
- int `main` (int argc, char \*\*argv)

## 13.286.1 Macro Definition Documentation

### 13.286.1.1 ENABLE\_ALL\_CHECKINGS

```

#define ENABLE_ALL_CHECKINGS 1

```

## 13.286.2 Function Documentation

### 13.286.2.1 check\_ftrstr()

```
template<typename Field , class RandIter >
bool check_ftrstr (
    const Field & F,
    size_t n,
    FFLAS::FFLAS_SIDE side,
    FFLAS::FFLAS_UPLO uplo,
    FFLAS::FFLAS_DIAG diagA,
    FFLAS::FFLAS_DIAG diagB,
    RandIter & Rand)
```

### 13.286.2.2 run\_with\_field()

```
template<class Field >
bool run_with_field (
    Givaro::Integer q,
    size_t b,
    size_t n,
    size_t iters,
    uint64_t seed)
```

### 13.286.2.3 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.287 test-fftrsv.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iomanip>
#include <iostream>
#include <random>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>
```

### Macros

- #define [\\_\\_FFLASFFPACK\\_SEQUENTIAL](#)
- #define [ENABLE\\_ALL\\_CHECKINGS](#) 1

### Functions

- template<typename [Field](#) , class RandIter >  
bool [check\\_fftrsv](#) (const [Field](#) &F, size\_t n, [FFLAS\\_UPLO](#) uplo, [FFLAS\\_TRANSPOSE](#) trans, [FFLAS\\_DIAG](#) diag, RandIter &Rand)
- template<class [Field](#) >  
bool [run\\_with\\_field](#) (Givaro::Integer q, size\_t b, size\_t n, size\_t iters, uint64\_t seed)
- int [main](#) (int argc, char \*\*argv)

## 13.287.1 Macro Definition Documentation

### 13.287.1.1 \_\_FFLASFFPACK\_SEQUENTIAL

```
#define __FFLASFFPACK_SEQUENTIAL
```

### 13.287.1.2 ENABLE\_ALL\_CHECKINGS

```
#define ENABLE_ALL_CHECKINGS 1
```

## 13.287.2 Function Documentation

### 13.287.2.1 check\_ftsrv()

```
template<typename Field , class RandIter >
bool check_ftsrv (
    const Field & F,
    size_t n,
    FFLAS_UPLO uplo,
    FFLAS_TRANSPOSE trans,
    FFLAS_DIAG diag,
    RandIter & Rand)
```

### 13.287.2.2 run\_with\_field()

```
template<class Field >
bool run_with_field (
    Givaro::Integer q,
    size_t b,
    size_t n,
    size_t iters,
    uint64_t seed)
```

### 13.287.2.3 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.288 test-fftrtri.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iomanip>
#include <iostream>
#include <random>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/ffpack/ffpack.h"
```

### Macros

- #define [\\_\\_FFLASFFPACK\\_SEQUENTIAL](#)
- #define [ENABLE\\_ALL\\_CHECKINGS](#) 1

## Functions

- template<typename [Field](#) , class RandIter >  
bool [check\\_ftrtri](#) (const [Field](#) &F, size\_t n, [FFLAS\\_UPLO](#) uplo, [FFLAS\\_DIAG](#) diag, RandIter &Rand)
- template<class [Field](#) >  
bool [run\\_with\\_field](#) (Givaro::Integer q, size\_t b, size\_t n, size\_t iters, uint64\_t seed)
- int [main](#) (int argc, char \*\*argv)

## 13.288.1 Macro Definition Documentation

### 13.288.1.1 [\\_\\_FFLASFFPACK\\_SEQUENTIAL](#)

```
#define __FFLASFFPACK_SEQUENTIAL
```

### 13.288.1.2 [ENABLE\\_ALL\\_CHECKINGS](#)

```
#define ENABLE_ALL_CHECKINGS 1
```

## 13.288.2 Function Documentation

### 13.288.2.1 [check\\_ftrtri\(\)](#)

```
template<typename Field , class RandIter >
bool check\_ftrtri (
    const Field & F,
    size_t n,
    FFLAS\_UPLO uplo,
    FFLAS\_DIAG diag,
    RandIter & Rand)
```

### 13.288.2.2 [run\\_with\\_field\(\)](#)

```
template<class Field >
bool run\_with\_field (
    Givaro::Integer q,
    size_t b,
    size_t n,
    size_t iters,
    uint64_t seed)
```

### 13.288.2.3 [main\(\)](#)

```
int main (
    int argc,
    char ** argv)
```

## 13.289 test-interfaces-c.c File Reference

```
#include <fflas-ffpack/interfaces/libs/fflas_c.h>
#include <fflas-ffpack/interfaces/libs/ffpack_c.h>
#include <stdlib.h>
#include <stdio.h>
```

## Functions

- int [main](#) ()

## 13.289.1 Function Documentation

### 13.289.1.1 main()

```
int main (  
    void )
```

## 13.290 test-invert-check.C File Reference

```
#include <iostream>  
#include <stdlib.h>  
#include "fflas-ffpack/fflas-ffpack.h"  
#include "fflas-ffpack/utils/args-parser.h"  
#include "fflas-ffpack/utils/test-utils.h"  
#include "fflas-ffpack/utils/fflas_randommatrix.h"
```

### Macros

- `#define` [ENABLE\\_ALL\\_CHECKINGS](#) 1

### Functions

- `int` [main](#) (`int` argc, `char **`argv)

## 13.290.1 Macro Definition Documentation

### 13.290.1.1 ENABLE\_ALL\_CHECKINGS

```
#define ENABLE_ALL_CHECKINGS 1
```

## 13.290.2 Function Documentation

### 13.290.2.1 main()

```
int main (  
    int argc,  
    char ** argv)
```

## 13.291 test-io.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"  
#include <iostream>  
#include <random>  
#include <givaro/modular.h>  
#include <givaro/zring.h>  
#include "fflas-ffpack/utils/test-utils.h"  
#include "fflas-ffpack/utils/fflas_io.h"  
#include "fflas-ffpack/utils/args-parser.h"
```

### Data Structures

- `struct` [CompactElement](#)< [Element](#) >
- `struct` [CompactElement](#)< `double` >
- `struct` [CompactElement](#)< `float` >
- `struct` [CompactElement](#)< `int64_t` >
- `struct` [CompactElement](#)< `int32_t` >
- `struct` [CompactElement](#)< `int16_t` >

## Functions

- template<class [Field](#) >  
bool [run\\_with\\_field](#) (Givaro::Integer q, uint64\_t b, size\_t m, size\_t n, size\_t iters, uint64\_t seed)
- int [main](#) (int argc, char \*\*argv)

### 13.291.1 Function Documentation

#### 13.291.1.1 [run\\_with\\_field\(\)](#)

```
template<class Field >
bool run_with_field (
    Givaro::Integer q,
    uint64_t b,
    size_t m,
    size_t n,
    size_t iters,
    uint64_t seed)
```

#### 13.291.1.2 [main\(\)](#)

```
int main (
    int argc,
    char ** argv)
```

## 13.292 test-lu.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <givaro/modular-balanced.h>
#include <iostream>
#include <iomanip>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/args-parser.h"
#include <random>
```

## Macros

- #define [BASECASE\\_K](#) 37
- #define [\\_\\_FFLASFFPACK\\_SEQUENTIAL](#)
- #define [\\_\\_LUDIVINE\\_CUTOFF](#) 1

## Functions

- template<class [Field](#) , [FFLAS\\_DIAG](#) diag, [FFLAS\\_TRANSPOSE](#) trans>  
bool [test\\_LUdivine](#) (const [Field](#) &F, typename [Field::ConstElement\\_ptr](#) A, size\_t lda, size\_t r, size\_t m, size\_t n)  
*Tests the LUdivine routine.*
- template<class [Field](#) , [FFLAS\\_DIAG](#) diag>  
bool [verifPLUQ](#) (const [Field](#) &F, typename [Field::ConstElement\\_ptr](#) A, size\_t lda, typename [Field::Element\\_ptr](#) PLUQ, size\_t ldpluq, size\_t \*P, size\_t \*Q, size\_t m, size\_t n, size\_t R)  
*Verifies that  $B = PLUQ$  where A stores  $[L|U]$ .*
- template<class [Field](#) , [FFLAS\\_DIAG](#) diag, class RandIter >  
bool [test\\_pluq](#) (const [Field](#) &F, typename [Field::ConstElement\\_ptr](#) A, size\_t r, size\_t m, size\_t n, size\_t lda, RandIter &G)

Tests the LUdivine routine.

- template<class [Field](#) , [FFLAS\\_DIAG](#) diag, [FFLAS\\_TRANSPOSE](#) trans, class Randlter >  
bool [launch\\_test](#) (const [Field](#) &F, size\_t r, size\_t m, size\_t n, Randlter &G)
- template<class [Field](#) >  
bool [run\\_with\\_field](#) (Givaro::Integer q, uint64\_t b, size\_t m, size\_t n, size\_t r, size\_t iters, uint64\_t seed)
- int [main](#) (int argc, char \*\*argv)

## Variables

- Givaro::Timer [tperm](#)
- Givaro::Timer [tgemm](#)
- Givaro::Timer [tBC](#)
- Givaro::Timer [ttrsm](#)
- Givaro::Timer [trest](#)
- Givaro::Timer [timtot](#)
- size\_t [mvcnt](#) = 0

## 13.292.1 Macro Definition Documentation

### 13.292.1.1 BASECASE\_K

```
#define BASECASE_K 37
```

### 13.292.1.2 \_\_FFLASFFPACK\_SEQUENTIAL

```
#define __FFLASFFPACK_SEQUENTIAL
```

### 13.292.1.3 \_\_LUDIVINE\_CUTOFF

```
#define __LUDIVINE_CUTOFF 1
```

## 13.292.2 Function Documentation

### 13.292.2.1 test\_LUdivine()

```
template<class Field , FFLAS::FFLAS\_DIAG diag, FFLAS\_TRANSPOSE trans>
bool test_LUdivine (
    const Field & F,
    typename Field::ConstElement\_ptr A,
    size_t lda,
    size_t r,
    size_t m,
    size_t n)
    Tests the LUdivine routine.
```

Tests the LUdivine routine.

#### Template Parameters

|              |                    |
|--------------|--------------------|
| <i>Field</i> | Field              |
| <i>Diag</i>  | Unit diagonal in U |
| <i>Trans</i> |                    |

#### Parameters

|          |                       |
|----------|-----------------------|
| <i>F</i> | field                 |
| <i>A</i> | Matrix (preallocated) |
| <i>r</i> | rank of A             |
| <i>m</i> | rows                  |

|            |                  |
|------------|------------------|
| <i>n</i>   | cols             |
| <i>lda</i> | leading dim of A |

**Returns**

0 iff correct, 1 otherwise

**13.292.2.2   verifPLUQ()**

```
template<class Field , FFLAS_DIAG diag>
bool verificPLUQ (
    const Field & F,
    typename Field::ConstElement_ptr A,
    size_t lda,
    typename Field::Element_ptr PLUQ,
    size_t ldpluq,
    size_t * P,
    size_t * Q,
    size_t m,
    size_t n,
    size_t R)

```

Verifies that  $B = PLUQ$  where A stores  $[L\backslash U]$ .

**Template Parameters**

|              |                    |
|--------------|--------------------|
| <i>Field</i> | Field              |
| <i>Diag</i>  | Unit diagonal in U |

**Parameters**

|            |                       |
|------------|-----------------------|
| <i>F</i>   | field                 |
| <i>A</i>   | Matrix (preallocated) |
| <i>r</i>   | rank of A             |
| <i>m</i>   | rows                  |
| <i>n</i>   | cols                  |
| <i>lda</i> | leading dim of A      |

**Returns**

0 iff correct, 1 otherwise

**13.292.2.3   test\_pluq()**

```
template<class Field , FFLAS_DIAG diag, class RandIter >
bool test_pluq (
    const Field & F,
    typename Field::ConstElement_ptr A,
    size_t r,
    size_t m,
    size_t n,
    size_t lda,
    RandIter & G)

```

Tests the LUdivine routine.



## Template Parameters

|              |                    |
|--------------|--------------------|
| <i>Field</i> | Field              |
| <i>Diag</i>  | Unit diagonal in U |
| <i>Trans</i> |                    |

## Parameters

|            |                       |
|------------|-----------------------|
| <i>F</i>   | field                 |
| <i>A</i>   | Matrix (preallocated) |
| <i>r</i>   | rank of A             |
| <i>m</i>   | rows                  |
| <i>n</i>   | cols                  |
| <i>lda</i> | leading dim of A      |

## Returns

0 iff correct, 1 otherwise

## 13.292.2.4 launch\_test()

```
template<class Field , FFLAS_DIAG diag, FFLAS_TRANSPOSE trans, class RandIter >
bool launch_test (
    const Field & F,
    size_t r,
    size_t m,
    size_t n,
    RandIter & G)
```

## 13.292.2.5 run\_with\_field()

```
template<class Field >
bool run_with_field (
    Givaro::Integer q,
    uint64_t b,
    size_t m,
    size_t n,
    size_t r,
    size_t iters,
    uint64_t seed)
```

## 13.292.2.6 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.292.3 Variable Documentation

## 13.292.3.1 tperm

Givaro::Timer tperm

## 13.292.3.2 tgemm

Givaro::Timer tgemm

**13.292.3.3 tBC**

```
Givaro::Timer tBC
```

**13.292.3.4 ttrsm**

```
Givaro::Timer ttrsm
```

**13.292.3.5 trest**

```
Givaro::Timer trest
```

**13.292.3.6 timtot**

```
Givaro::Timer timtot
```

**13.292.3.7 mvcnt**

```
size_t mvcnt = 0
```

**13.293 test-maxdelayeddim.C File Reference**

```
#include <givaro/modular.h>
#include <recint/rint.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include <stdlib.h>
#include <stdio.h>
```

**Macros**

- #define [MAX\\_WITH\\_SIZE\\_T](#)(x)

**Functions**

- template<class [Field](#) >  
bool [test](#) (Givaro::Integer p, size\_t kmax)
- int [main](#) ()

**13.293.1 Macro Definition Documentation****13.293.1.1 MAX\_WITH\_SIZE\_T**

```
#define MAX_WITH_SIZE_T(  
    x)
```

**Value:**

```
( (static_cast<uint64_t>(std::numeric_limits<size_t>::max()) < x)? std::numeric_limits<size_t>::max() : x )
```

**13.293.2 Function Documentation****13.293.2.1 test()**

```
template<class Field >
bool test (
    Givaro::Integer p,
    size_t kmax)
```

**13.293.2.2 main()**

```
int main (
    void )
```

## 13.294 test-minpoly.C File Reference

```
#include <iomanip>
#include <iostream>
#include <random>
#include <chrono>
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/ffpack/args-parser.h"
#include "fflas-ffpack/ffpack/fflas_randommatrix.h"
#include "fflas-ffpack/ffpack/test-utils.h"
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
#include <givaro/modular-integer.h>
#include <givaro/givpoly1factor.h>
#include <givaro/givpoly1.h>
```

### Functions

- template<typename [Field](#) , class RandIter >  
bool [check\\_minpoly](#) (const [Field](#) &F, size\_t n, RandIter &G)
- template<class [Field](#) >  
bool [run\\_with\\_field](#) (Givaro::Integer q, size\_t b, size\_t n, size\_t iters, uint64\_t seed)
- int [main](#) (int argc, char \*\*argv)

### 13.294.1 Function Documentation

#### 13.294.1.1 [check\\_minpoly\(\)](#)

```
template<typename Field , class RandIter >
bool check\_minpoly (
    const Field & F,
    size_t n,
    RandIter & G)
```

#### 13.294.1.2 [run\\_with\\_field\(\)](#)

```
template<class Field >
bool run\_with\_field (
    Givaro::Integer q,
    size_t b,
    size_t n,
    size_t iters,
    uint64_t seed)
```

#### 13.294.1.3 [main\(\)](#)

```
int main (
    int argc,
    char ** argv)
```

## 13.295 test-multifile1.C File Reference

```
#include "fflas-ffpack/fflas-ffpack.h"
```

## 13.296 test-multifile2.C File Reference

```
#include "fflas-ffpack/fflas-ffpack.h"
```

### Functions

- int [main](#) (void)

### 13.296.1 Function Documentation

#### 13.296.1.1 main()

```
int main (
    void )
```

## 13.297 test-nullspace.C File Reference

```
#include <iomanip>
#include <iostream>
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/timer.h"
```

### Functions

- template<class [Field](#) >  
std::string [checkingMessage](#) (const [Field](#) &F)
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [readOrRandomMatrixWithRankAndRandomRPM](#) (const [Field](#) &F, std::string file, size\_t m, size\_t n, size\_t lda, size\_t r, uint64\_t seed)  
*If file is not empty, read it and set m, n, lda and r.*
- template<class [Field](#) >  
bool [test\\_nullspace](#) ([Field](#) &F, [FFLAS::FFLAS\\_SIDE](#) side, size\_t m, size\_t n, size\_t r, typename [Field::Element\\_ptr](#) A, size\_t lda)
- template<class [Field](#) >  
bool [run\\_with\\_field](#) (Givaro::Integer q, uint64\_t b, size\_t m, size\_t n, size\_t r, size\_t iters, std::string file, uint64\_t &seed)
- int [main](#) (int argc, char \*\*argv)

### 13.297.1 Function Documentation

#### 13.297.1.1 checkingMessage()

```
template<class Field >
std::string checkingMessage (
    const Field & F)
```

#### 13.297.1.2 readOrRandomMatrixWithRankAndRandomRPM()

```
template<class Field >
Field::Element\_ptr readOrRandomMatrixWithRankAndRandomRPM (
    const Field & F,
    std::string file,
```

```

    size_t & m,
    size_t & n,
    size_t & lda,
    size_t & r,
    uint64_t seed)

```

If file is not empty, read it and set m, n, lda and r.

Otherwise, generate a random matrix of size m x n with random lda.

### 13.297.1.3 test\_nullspace()

```

template<class Field >
bool test_nullspace (
    Field & F,
    FFLAS::FFLAS_SIDE side,
    size_t m,
    size_t n,
    size_t r,
    typename Field::Element_ptr A,
    size_t lda)

```

### 13.297.1.4 run\_with\_field()

```

template<class Field >
bool run_with_field (
    Givaro::Integer q,
    uint64_t b,
    size_t m,
    size_t n,
    size_t r,
    size_t iters,
    std::string file,
    uint64_t & seed)

```

### 13.297.1.5 main()

```

int main (
    int argc,
    char ** argv)

```

## 13.298 test-permutations.C File Reference

```

#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include "fflas-ffpack/ffpack/ffpack.h"

```

### Functions

- bool [checkMonotonicApplyP](#) (FFLAS\_SIDE Side, FFLAS\_TRANSPOSE trans, size\_t \*P, size\_t N, size\_t R)
- int [main](#) ()

### Variables

- Givaro::Timer [tperm](#)
- Givaro::Timer [tgemm](#)
- Givaro::Timer [tBC](#)
- Givaro::Timer [ttrsm](#)

- Givaro::Timer [trest](#)
- Givaro::Timer [timtot](#)

## 13.298.1 Function Documentation

### 13.298.1.1 checkMonotonicApplyP()

```
bool checkMonotonicApplyP (
    FFLAS_SIDE Side,
    FFLAS_TRANSPOSE trans,
    size_t * P,
    size_t N,
    size_t R)
```

### 13.298.1.2 main()

```
int main (
    void )
```

## 13.298.2 Variable Documentation

### 13.298.2.1 tperm

```
Givaro::Timer tperm
```

### 13.298.2.2 tgemm

```
Givaro::Timer tgemm
```

### 13.298.2.3 tBC

```
Givaro::Timer tBC
```

### 13.298.2.4 ttrsm

```
Givaro::Timer ttrsm
```

### 13.298.2.5 trest

```
Givaro::Timer trest
```

### 13.298.2.6 timtot

```
Givaro::Timer timtot
```

## 13.299 test-pluq-check.C File Reference

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
```

### Macros

- #define [ENABLE\\_ALL\\_CHECKINGS](#) 1

## Functions

- int [main](#) (int argc, char \*\*argv)

### 13.299.1 Macro Definition Documentation

#### 13.299.1.1 ENABLE\_ALL\_CHECKINGS

```
#define ENABLE_ALL_CHECKINGS 1
```

### 13.299.2 Function Documentation

#### 13.299.2.1 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.300 test-quasisep.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <givaro/modular-balanced.h>
#include <iostream>
#include <iomanip>
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/args-parser.h"
#include <random>
```

## Functions

- template<class [Field](#) , [FFLAS\\_DIAG](#) diag, class RandIter >  
bool [test\\_BruhatGenerator](#) (const [Field](#) &F, size\_t n, size\_t r, size\_t t, typename [Field::ConstElement\\_ptr](#) A, size\_t lda, typename [Field::Element\\_ptr](#) TS, size\_t l, RandIter &G)
- template<class [Field](#) , [FFLAS\\_DIAG](#) diag, class RandIter >  
bool [launch\\_test](#) (const [Field](#) &F, size\_t n, size\_t r, size\_t t, size\_t l, RandIter &G)
- template<class [Field](#) , class RandGen >  
bool [testLTQSRPM](#) (const [Field](#) &F, size\_t n, size\_t r, size\_t t, RandGen &G)
- template<class [Field](#) >  
bool [run\\_with\\_field](#) (Givaro::Integer q, uint64\_t b, size\_t n, size\_t r, size\_t t, size\_t l, size\_t iters, uint64\_t seed)
- int [main](#) (int argc, char \*\*argv)

### 13.300.1 Function Documentation

#### 13.300.1.1 test\_BruhatGenerator()

```
template<class Field , FFLAS\_DIAG diag, class RandIter >
bool test_BruhatGenerator (
    const Field & F,
    size_t n,
    size_t r,
    size_t t,
    typename Field::ConstElement\_ptr A,
    size_t lda,
    typename Field::Element\_ptr TS,
    size_t l,
    RandIter & G)
```

**13.300.1.2 launch\_test()**

```
template<class Field , FFLAS_DIAG diag, class RandIter >
bool launch_test (
    const Field & F,
    size_t n,
    size_t r,
    size_t t,
    size_t l,
    RandIter & G)
```

**13.300.1.3 testLTQSRPM()**

```
template<class Field , class RandGen >
bool testLTQSRPM (
    const Field & F,
    size_t n,
    size_t r,
    size_t t,
    RandGen & G)
```

**13.300.1.4 run\_with\_field()**

```
template<class Field >
bool run_with_field (
    Givaro::Integer q,
    uint64_t b,
    size_t n,
    size_t r,
    size_t t,
    size_t l,
    size_t iters,
    uint64_t seed)
```

**13.300.1.5 main()**

```
int main (
    int argc,
    char ** argv)
```

**13.301 test-rankprofiles.C File Reference**

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>
#include <iostream>
#include <iomanip>
#include <random>
#include <chrono>
```

**Macros**

- #define `__FFLASFFPACK_SEQUENTIAL`



## Functions

- template<class [Field](#) >  
bool [run\\_with\\_field](#) (Givaro::Integer q, uint64\_t b, size\_t m, size\_t n, size\_t r, size\_t iters, uint64\_t seed, bool par)
- int [main](#) (int argc, char \*\*argv)

### 13.301.1 Macro Definition Documentation

#### 13.301.1.1 \_\_FFLASFFPACK\_SEQUENTIAL

```
#define __FFLASFFPACK_SEQUENTIAL
```

### 13.301.2 Function Documentation

#### 13.301.2.1 run\_with\_field()

```
template<class Field >
bool run_with_field (
    Givaro::Integer q,
    uint64_t b,
    size_t m,
    size_t n,
    size_t r,
    size_t iters,
    uint64_t seed,
    bool par)
```

#### 13.301.2.2 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.302 test-rpm.C File Reference

```
#include <iostream>
#include "givaro/modular.h"
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/ffpack/ffpack.h"
```

## Functions

- bool [checkRPM](#) (size\_t M, size\_t N, size\_t R)
- bool [checkSymmetricRPM](#) (size\_t N, size\_t R)
- int [main](#) (int argc, char \*\*argv)

### 13.302.1 Function Documentation

#### 13.302.1.1 checkRPM()

```
bool checkRPM (
    size_t M,
    size_t N,
    size_t R)
```

**13.302.1.2 checkSymmetricRPM()**

```
bool checkSymmetricRPM (
    size_t N,
    size_t R)
```

**13.302.1.3 main()**

```
int main (
    int argc,
    char ** argv)
```

**13.303 test-simd.C File Reference**

```
#include "givaro/givinteger.h"
#include "givaro/modular.h"
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/fflas/fflas_simd.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include <array>
#include <vector>
#include <random>
#include <string>
#include <functional>
#include <limits>
#include <type_traits>
#include <algorithm>
```

**Data Structures**

- struct [ALL< true, v... >](#)
- struct [ALL< false, v... >](#)
- struct [ALL<>](#)
- struct [count\\_nonconst\\_lvalue\\_reference< T, O... >](#)
- struct [count\\_nonconst\\_lvalue\\_reference< T &, O... >](#)
- struct [count\\_nonconst\\_lvalue\\_reference< const T &, O... >](#)
- struct [count\\_nonconst\\_lvalue\\_reference<>](#)
- struct [is\\_all\\_same< T, Args... >](#)
- struct [is\\_all\\_same<>](#)
- struct [width< T >](#)
- struct [width< float >](#)
- struct [width< double >](#)
- class [TestOneMethod< Simd >](#)
- struct [ScalFunctionsBase< Element, typename enable\\_if< is\\_floating\\_point< Element >::value >::type >](#)
- class [ScalFunctionsBase< Element, typename enable\\_if< is\\_floating\\_point< Element >::value >::type >::FloatingPointTestD](#)
- struct [ScalFunctionsBase< Element, typename enable\\_if< is\\_integral< Element >::value >::type >](#)
- struct [ScalFunctions< Element >](#)

**Macros**

- [#define \\_TEST\\_ONE\(K, f1, f2, r, n\)](#)
- [#define TEST\\_ONE\\_OP\(f\)](#)
- [#define TEST\\_ONE\\_OP\\_WZ\(f\)](#)
- [#define TEST\\_IMPL\(SIZE, Elt\)](#)

## Functions

- `template<typename Element >`  
`enable_if< is_integral< Element >::value, bool >::type` `check_eq` (Element x, Element y)
- `template<typename Element >`  
`enable_if< is_floating_point< Element >::value, bool >::type` `check_eq` (Element x, Element y)
- `template<typename Element >`  
`bool` `cmp` (vector< Element > out\_scal, vector< Element > out\_simd)
- `template<typename Ret , typename T >`  
`Ret` `eval_func_on_array` (function< Ret()> f, array< T, 0 > &arr)
- `template<typename T , typename... TArgs>`  
`void` `eval_func_on_array` (function< void(T, TArgs...)> f, array< typename decay< T >::type, sizeof...(TArgs)+1 > &arr)
- `template<typename Ret , typename T , typename... TArgs>`  
`Ret` `eval_func_on_array` (function< Ret(T, TArgs...)> f, array< typename decay< T >::type, sizeof...(TArgs)+1 > &arr)
- `template<typename E >`  
`std::ostream &` `operator<<` (std::ostream &o, const vector< E > &V)
- `template<typename Simd , typename Element >`  
`enable_if< is_floating_point< Element >::value, bool >::type` `test_impl_base` ()
- `template<typename Simd , typename Element >`  
`enable_if< is_integral< Element >::value, bool >::type` `test_impl_base` ()
- `template<typename Simd , typename Element >`  
`bool` `test_impl` ()
- `int` `main` (int argc, char \*argv[])

### 13.303.1 Macro Definition Documentation

#### 13.303.1.1 \_TEST\_ONE

```
#define _TEST_ONE(
```

```
    K,  
    f1,  
    f2,  
    r,  
    n)
```

Value:

```
do {  
    K T(f1, f2, r, n);  
    bool b = T.writeResultLine();  
    if (b == false)  
        T.writeDebugData();  
    btest &= b;  
} while (0)
```

#### 13.303.1.2 TEST\_ONE\_OP

```
#define TEST_ONE_OP(
```

```
    f)
```

Value:

```
_TEST_ONE(TestOneMethod<Simd>,  
function<decltype(Simd::f)>(Simd::f),  
function<decltype(Scal::f)>(Scal::f),  
function<decltype(Scal::genInputs)>(Scal::genInputs), #f)
```

#### 13.303.1.3 TEST\_ONE\_OP\_WZ

```
#define TEST_ONE_OP_WZ(
```

```
    f)
```

Value:

```
_TEST_ONE(TestOneMethod<Simd>,  
function<decltype(Simd::f)>(Simd::f),  
function<decltype(Scal::f)>(Scal::f),  
function<decltype(Scal::genInputsWithZero)>(Scal::genInputsWithZero),  
#f " test with zero")
```

### 13.303.1.4 TEST\_IMPL

```
#define TEST_IMPL(  
    SIZE,  
    Elt)
```

Value:

```
do {  
    pass &= test_impl<Simd##SIZE<Elt>, Elt>(); \  
    cout << endl; \  
} while (0)
```

## 13.303.2 Function Documentation

### 13.303.2.1 check\_eq() [1/2]

```
template<typename Element >  
enable_if< is_integral< Element >::value, bool >::type check_eq (  
    Element x,  
    Element y)
```

### 13.303.2.2 check\_eq() [2/2]

```
template<typename Element >  
enable_if< is_floating_point< Element >::value, bool >::type check_eq (  
    Element x,  
    Element y)
```

### 13.303.2.3 cmp()

```
template<typename Element >  
bool cmp (  
    vector< Element > out_scal,  
    vector< Element > out_simd)
```

### 13.303.2.4 eval\_func\_on\_array() [1/3]

```
template<typename Ret , typename T >  
Ret eval_func_on_array (  
    function< Ret()> f,  
    array< T, 0 > & arr)
```

### 13.303.2.5 eval\_func\_on\_array() [2/3]

```
template<typename T , typename... TArgs>  
void eval_func_on_array (  
    function< void(T, TArgs...)> f,  
    array< typename decay< T >::type, sizeof...(TArgs)+1 > & arr)
```

### 13.303.2.6 eval\_func\_on\_array() [3/3]

```
template<typename Ret , typename T , typename... TArgs>  
Ret eval_func_on_array (  
    function< Ret(T, TArgs...)> f,  
    array< typename decay< T >::type, sizeof...(TArgs)+1 > & arr)
```

### 13.303.2.7 operator<<()

```
template<typename E >  
std::ostream & operator<< (  
    std::ostream & o,  
    const vector< E > & V)
```

**13.303.2.8 test\_impl\_base() [1/2]**

```
template<typename Simd , typename Element >
enable_if< is_floating_point< Element >::value, bool >::type test_impl_base ()
```

**13.303.2.9 test\_impl\_base() [2/2]**

```
template<typename Simd , typename Element >
enable_if< is_integral< Element >::value, bool >::type test_impl_base ()
```

**13.303.2.10 test\_impl()**

```
template<typename Simd , typename Element >
bool test_impl ()
```

**13.303.2.11 main()**

```
int main (
    int argc,
    char * argv[])
```

**13.304 test-solve.C File Reference**

```
#include <givaro/modular-integer.h>
#include <iomanip>
#include <iostream>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
```

**Functions**

- template<typename [Field](#) , class RandIter >  
bool [check\\_solve](#) (const [Field](#) &F, size\_t m, RandIter &Rand, bool isParallel)
- template<class [Field](#) >  
bool [run\\_with\\_field](#) (Givaro::Integer q, size\_t b, size\_t m, size\_t iters, uint64\_t seed)
- int [main](#) (int argc, char \*\*argv)

**13.304.1 Function Documentation****13.304.1.1 check\_solve()**

```
template<typename Field , class RandIter >
bool check_solve (
    const Field & F,
    size_t m,
    RandIter & Rand,
    bool isParallel)
```

**13.304.1.2 run\_with\_field()**

```
template<class Field >
bool run_with_field (
    Givaro::Integer q,
    size_t b,
```

```

    size_t m,
    size_t iters,
    uint64_t seed)

```

### 13.304.1.3 main()

```

int main (
    int argc,
    char ** argv)

```

## 13.305 test-storage-transpose.C File Reference

```

#include <iomanip>
#include <iostream>
#include <random>
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/fflas_memory.h"
#include <givaro/modular.h>
#include <recint/rint.h>
#include "fflas-ffpack/fflas/fflas_transpose.h"

```

### Data Structures

- class [Test< Elt >](#)

### Functions

- int [main](#) (int argc, char \*\*argv)

## 13.305.1 Function Documentation

### 13.305.1.1 main()

```

int main (
    int argc,
    char ** argv)

```

## 13.306 101-fgemv.C File Reference

```

#include <fflas-ffpack/fflas-ffpack-config.h>
#include <givaro/modular-balanced.h>
#include <fflas-ffpack/fflas/fflas.h>
#include <fflas-ffpack/utils/timer.h>
#include <fflas-ffpack/utils/fflas_io.h>
#include <fflas-ffpack/utils/args-parser.h>
#include <iostream>

```

### Functions

- int [main](#) (int argc, char \*\*argv)

## 13.306.1 Function Documentation

### 13.306.1.1 main()

```
int main (  
    int argc,  
    char ** argv)
```

## 13.307 2x2-fgemm.C File Reference

```
#include <fflas-ffpack/fflas-ffpack-config.h>  
#include <givaro/modular-balanced.h>  
#include <fflas-ffpack/fflas/fflas.h>  
#include <fflas-ffpack/utils/timer.h>  
#include <fflas-ffpack/utils/fflas_io.h>  
#include <fflas-ffpack/utils/args-parser.h>  
#include <iostream>
```

### Functions

- int [main](#) (int argc, char \*\*argv)

## 13.307.1 Function Documentation

### 13.307.1.1 main()

```
int main (  
    int argc,  
    char ** argv)
```

## 13.308 2x2-ftsrv.C File Reference

```
#include <fflas-ffpack/fflas-ffpack-config.h>  
#include <givaro/modular-balanced.h>  
#include <fflas-ffpack/fflas/fflas.h>  
#include <fflas-ffpack/utils/timer.h>  
#include <fflas-ffpack/utils/fflas_io.h>  
#include <fflas-ffpack/utils/args-parser.h>  
#include <iostream>  
#include <array>
```

### Functions

- int [main](#) (int argc, char \*\*argv)

## 13.308.1 Function Documentation

### 13.308.1.1 main()

```
int main (  
    int argc,  
    char ** argv)
```

## 13.309 2x2-pluq.C File Reference

```
#include <iostream>
#include <vector>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/fflas_io.h"
```

### Functions

- int [main](#) (int argc, char \*\*argv)

### 13.309.1 Function Documentation

#### 13.309.1.1 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.310 fflas-101\_1.C File Reference

```
#include <fflas-ffpack/fflas/fflas.h>
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/fflas_io.h"
#include <iostream>
```

### Functions

- int [main](#) (int argc, char \*\*argv)

### 13.310.1 Function Documentation

#### 13.310.1.1 main()

```
int main (
    int argc,
    char ** argv)
```

## 13.311 fflas-101\_3.C File Reference

```
#include <fflas-ffpack/fflas/fflas.h>
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include <iostream>
```

### Functions

- int [main](#) (int argc, char \*\*argv)



### 13.311.1 Function Documentation

#### 13.311.1.1 main()

```
int main (  
    int argc,  
    char ** argv)
```

## 13.312 fflas\_101.C File Reference

```
#include <fflas-ffpack/fflas/fflas.h>  
#include <givaro/modular.h>  
#include <givaro/modular-balanced.h>  
#include <iostream>
```

### Functions

- int [main](#) (int argc, char \*\*argv)

### 13.312.1 Function Documentation

#### 13.312.1.1 main()

```
int main (  
    int argc,  
    char ** argv)
```

## 13.313 fflas\_101\_lvl1.C File Reference

```
#include <fflas-ffpack/fflas/fflas.h>  
#include <givaro/modular.h>  
#include <givaro/modular-balanced.h>  
#include <iostream>  
#include "fflas-ffpack/utils/fflas_io.h"
```

### Functions

- int [main](#) (int argc, char \*\*argv)

### 13.313.1 Function Documentation

#### 13.313.1.1 main()

```
int main (  
    int argc,  
    char ** argv)
```

## 13.314 ffpack-fgesv.C File Reference

```
#include <fflas-ffpack/fflas/fflas.h>  
#include <givaro/modular.h>  
#include <givaro/modular-balanced.h>  
#include "fflas-ffpack/utils/fflas_io.h"  
#include <fflas-ffpack/ffpack/ffpack.h>  
#include <iostream>
```

## Functions

- int [main](#) (int argc, char \*\*argv)

### 13.314.1 Function Documentation

#### 13.314.1.1 main()

```
int main (  
    int argc,  
    char ** argv)
```

## 13.315 ffpack-solve.C File Reference

```
#include <fflas-ffpack/fflas/fflas.h>  
#include <givaro/modular.h>  
#include <givaro/modular-balanced.h>  
#include "fflas-ffpack/utils/fflas_io.h"  
#include <fflas-ffpack/ffpack/ffpack.h>  
#include <iostream>
```

## Functions

- int [main](#) (int argc, char \*\*argv)

### 13.315.1 Function Documentation

#### 13.315.1.1 main()

```
int main (  
    int argc,  
    char ** argv)
```

PS: the function Solve will modify the matrix A so here we used a duplicate matrix A2 otherwise A\*x will not be equal to b for the later verification stage

# Index

- [\\_F](#)
  - [RNSIntegerMod< RNS >, 551](#)
- [\\_M](#)
  - [rns\\_double, 530](#)
  - [rns\\_double\\_extended, 541](#)
- [\\_MAX\\_SIZE\\_MATRICES](#)
  - [benchmark-checkers.C, 781](#)
- [\\_MMi](#)
  - [rns\\_double, 531](#)
  - [rns\\_double\\_extended, 541](#)
- [\\_Mi](#)
  - [rns\\_double, 530](#)
  - [rns\\_double\\_extended, 541](#)
- [\\_Mi\\_modp\\_rns](#)
  - [RNSIntegerMod< RNS >, 551](#)
- [\\_NR\\_TESTS](#)
  - [benchmark-checkers.C, 781](#)
- [\\_PLUQ](#)
  - [FFPACK, 363](#)
- [\\_RNSdelayed](#)
  - [RNSIntegerMod< RNS >, 552](#)
- [\\_TEST\\_ONE](#)
  - [test-simd.C, 1101](#)
- [\\_\\_FFLASFFPACK\\_ARITHPROG\\_THRESHOLD](#)
  - [fflas-ffpack-default-thresholds.h, 825](#)
- [\\_\\_FFLASFFPACK\\_CACHE\\_LINE\\_SIZE](#)
  - [fflas\\_sparse.h, 885](#)
- [\\_\\_FFLASFFPACK\\_CHARPOLY\\_Danilevskii\\_LUKrylov\\_THRESHOLD](#)
  - [fflas-ffpack-default-thresholds.h, 824](#)
- [\\_\\_FFLASFFPACK\\_CHARPOLY\\_LUKrylov\\_ArithProg\\_THRESHOLD](#)
  - [fflas-ffpack-default-thresholds.h, 824](#)
- [\\_\\_FFLASFFPACK\\_CONFIGURATION](#)
  - [cblas.C, 1051](#)
  - [clapack.C, 1051](#)
  - [fblas.C, 1052](#)
  - [lapack.C, 1053](#)
- [\\_\\_FFLASFFPACK\\_DIMKPENALTY](#)
  - [fflas\\_pfgemm.inl, 870](#)
- [\\_\\_FFLASFFPACK\\_FORCE\\_SEQ](#)
  - [benchmark-charpoly-mp.C, 779](#)
- [\\_\\_FFLASFFPACK\\_FSYRK\\_THRESHOLD](#)
  - [fflas-ffpack-default-thresholds.h, 825](#)
- [\\_\\_FFLASFFPACK\\_FSYTRF\\_THRESHOLD](#)
  - [fflas-ffpack-default-thresholds.h, 825](#)
- [\\_\\_FFLASFFPACK\\_FTRSSYR2K\\_THRESHOLD](#)
  - [ffpack.h, 923](#)
- [\\_\\_FFLASFFPACK\\_FTRSTR\\_THRESHOLD](#)
  - [ffpack.h, 923](#)
- [\\_\\_FFLASFFPACK\\_FTRTRI\\_THRESHOLD](#)
  - [fflas-ffpack-default-thresholds.h, 825](#)
- [\\_\\_FFLASFFPACK\\_GAUSSJORDAN\\_BASECASE](#)
  - [ffpack\\_echelonforms.inl, 931](#)
  - [test-echelon.C, 1059](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_BLAS](#)
  - [config.h, 821](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_CBLAS](#)
  - [cblas.C, 1051](#)
  - [config.h, 821](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_CLAPACK](#)
  - [clapack.C, 1051](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_CXX11](#)
  - [config.h, 821](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_DGETRF](#)
  - [benchmark-dgetrf.C, 782](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_DLFCN\\_H](#)
  - [config.h, 821](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_DTRTRI](#)
  - [benchmark-dtrtri.C, 785](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_FLOAT\\_H](#)
  - [config.h, 821](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_INT128](#)
  - [config.h, 821](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_INTPTR\\_T](#)
  - [config.h, 821](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_INTTYPES\\_H](#)
  - [config.h, 821](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_LAPACK](#)
  - [clapack.C, 1051](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_LIMITS\\_H](#)
  - [config.h, 821](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_LITTLE\\_ENDIAN](#)
  - [config.h, 821](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_PTHREAD\\_H](#)
  - [config.h, 821](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_STDDEF\\_H](#)
  - [config.h, 821](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_STDINT\\_H](#)
  - [config.h, 822](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_STDIO\\_H](#)
  - [config.h, 822](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_STDLIB\\_H](#)
  - [config.h, 822](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_STRINGS\\_H](#)
  - [config.h, 822](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_STRING\\_H](#)
  - [config.h, 822](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_SYS\\_STAT\\_H](#)
  - [config.h, 822](#)

- \_\_FFLASFFPACK\_HAVE\_SYS\_TIME\_H
  - config.h, [822](#)
- \_\_FFLASFFPACK\_HAVE\_SYS\_TYPES\_H
  - config.h, [822](#)
- \_\_FFLASFFPACK\_HAVE\_UNISTD\_H
  - config.h, [822](#)
- \_\_FFLASFFPACK\_KAAPI\_ROUTINES\_INL
  - kaapi\_routines.inl, [1032](#)
- \_\_FFLASFFPACK\_LT\_OBJDIR
  - config.h, [822](#)
- \_\_FFLASFFPACK\_MINBLOCKCUTS
  - blockcuts.inl, [1031](#)
- \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET
  - benchmark-charpoly.C, [780](#)
  - benchmark-fadd-lvl2.C, [785](#)
  - benchmark-fdot.C, [786](#)
  - benchmark-fgemm-mp.C, [787](#)
  - benchmark-fgemm-rns.C, [788](#)
  - benchmark-fgemv-mp.C, [790](#)
  - benchmark-fgemv.C, [791](#)
  - benchmark-fgesv.C, [794](#)
  - benchmark-fsyrc.C, [795](#)
  - benchmark-fsytrf.C, [796](#)
  - benchmark-ftrsm-mp.C, [796](#)
  - benchmark-ftrsm.C, [797](#)
  - benchmark-ftrsv.C, [798](#)
  - benchmark-ftrtri.C, [798](#)
  - benchmark-pluq.C, [801](#)
  - benchmark-quasisep.C, [802](#)
- \_\_FFLASFFPACK\_OPENBLAS\_NUM\_THREADS
  - config.h, [822](#)
- \_\_FFLASFFPACK\_PACKAGE
  - config.h, [822](#)
- \_\_FFLASFFPACK\_PACKAGE\_BUGREPORT
  - config.h, [822](#)
- \_\_FFLASFFPACK\_PACKAGE\_NAME
  - config.h, [822](#)
- \_\_FFLASFFPACK\_PACKAGE\_STRING
  - config.h, [822](#)
- \_\_FFLASFFPACK\_PACKAGE\_TARNAME
  - config.h, [823](#)
- \_\_FFLASFFPACK\_PACKAGE\_URL
  - config.h, [823](#)
- \_\_FFLASFFPACK\_PACKAGE\_VERSION
  - config.h, [823](#)
- \_\_FFLASFFPACK\_PLUQ\_THRESHOLD
  - fflas-ffpack-default-thresholds.h, [824](#)
  - test-echelon.C, [1059](#)
- \_\_FFLASFFPACK\_SEQPARTHRESHOLD
  - fflas\_pfgemm.inl, [870](#)
- \_\_FFLASFFPACK\_SEQUENTIAL
  - parallel.h, [1033](#)
  - test-echelon.C, [1059](#)
  - test-ftrmm.C, [1078](#)
  - test-ftrmv.C, [1079](#)
  - test-ftrsm.C, [1081](#)
  - test-ftrsv.C, [1085](#)
  - test-ftrtri.C, [1086](#)
  - test-lu.C, [1089](#)
  - test-rankprofiles.C, [1099](#)
- \_\_FFLASFFPACK\_SIZEOF\_CHAR
  - config.h, [823](#)
- \_\_FFLASFFPACK\_SIZEOF\_INT
  - config.h, [823](#)
- \_\_FFLASFFPACK\_SIZEOF\_LONG
  - config.h, [823](#)
- \_\_FFLASFFPACK\_SIZEOF\_LONG\_LONG
  - config.h, [823](#)
- \_\_FFLASFFPACK\_SIZEOF\_SHORT
  - config.h, [823](#)
- \_\_FFLASFFPACK\_SIZEOF\_\_\_INT64\_T
  - config.h, [823](#)
- \_\_FFLASFFPACK\_STDC\_HEADERS
  - config.h, [823](#)
- \_\_FFLASFFPACK\_USE\_OPENMP
  - config.h, [823](#)
- \_\_FFLASFFPACK\_VERSION
  - config.h, [823](#)
- \_\_FFLASFFPACK\_WINOTHRESHOLD
  - fflas-ffpack-default-thresholds.h, [824](#)
- \_\_FFLASFFPACK\_WINOTHRESHOLD\_BAL
  - fflas-ffpack-default-thresholds.h, [824](#)
- \_\_FFLASFFPACK\_WINOTHRESHOLD\_BAL\_FLT
  - fflas-ffpack-default-thresholds.h, [824](#)
- \_\_FFLASFFPACK\_WINOTHRESHOLD\_FLT
  - fflas-ffpack-default-thresholds.h, [824](#)
- \_\_FFLASFFPACK\_charpoly\_INL
  - ffpack\_charpoly.inl, [927](#)
- \_\_FFLASFFPACK\_checker\_charpoly\_INL
  - checker\_charpoly.inl, [806](#)
- \_\_FFLASFFPACK\_checker\_det\_INL
  - checker\_det.inl, [806](#)
- \_\_FFLASFFPACK\_checker\_fgemm\_INL
  - checker\_fgemm.inl, [807](#)
- \_\_FFLASFFPACK\_checker\_ftrsm\_INL
  - checker\_ftrsm.inl, [807](#)
- \_\_FFLASFFPACK\_checker\_invert\_INL
  - checker\_invert.inl, [807](#)
- \_\_FFLASFFPACK\_checker\_pluq\_INL
  - checker\_pluq.inl, [808](#)
- \_\_FFLASFFPACK\_fadd\_INL
  - fflas\_fadd.inl, [830](#)
- \_\_FFLASFFPACK\_fassign\_INL
  - fflas\_fassign.inl, [831](#)
- \_\_FFLASFFPACK\_faxpy\_INL
  - fflas\_faxpy.inl, [832](#)
- \_\_FFLASFFPACK\_fdot\_INL
  - fflas\_fdot.inl, [833](#)
- \_\_FFLASFFPACK\_fflas\_blockcuts\_INL
  - blockcuts.inl, [1031](#)
- \_\_FFLASFFPACK\_fflas\_bounds\_INL
  - fflas\_bounds.inl, [827](#)
- \_\_FFLASFFPACK\_fflas\_fflas\_fgemm\_winograd\_INL
  - fgemm\_winograd.inl, [839](#)
- \_\_FFLASFFPACK\_fflas\_fflas\_level1\_INL
  - fflas\_level1.inl, [864](#)

- \_\_FFLASFFPACK\_fflas\_fflas\_level2\_INL  
fflas\_level2.inl, [867](#)
- \_\_FFLASFFPACK\_fflas\_fflas\_level3\_INL  
fflas\_level3.inl, [870](#)
- \_\_FFLASFFPACK\_fflas\_fflas\_mmhelper\_INL  
fflas\_helpers.inl, [859](#)
- \_\_FFLASFFPACK\_fflas\_fflas\_sparse\_INL  
fflas\_sparse.inl, [887](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_INL  
simd128.inl, [873](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_double\_INL  
simd128\_double.inl, [874](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_float\_INL  
simd128\_float.inl, [874](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_int16\_INL  
simd128\_int16.inl, [875](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_int32\_INL  
simd128\_int32.inl, [875](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_int64\_INL  
simd128\_int64.inl, [876](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_INL  
simd256.inl, [876](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_double\_INL  
simd256\_double.inl, [877](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_float\_INL  
simd256\_float.inl, [877](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_int16\_INL  
simd256\_int16.inl, [878](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_int32\_INL  
simd256\_int32.inl, [878](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_int64\_INL  
simd256\_int64.inl, [878](#)
- \_\_FFLASFFPACK\_fflas\_freduce\_INL  
fflas\_freduce.inl, [849](#)
- \_\_FFLASFFPACK\_fflas\_freduce\_mp\_INL  
fflas\_freduce\_mp.inl, [849](#)
- \_\_FFLASFFPACK\_fflas\_fsy2k\_INL  
fflas\_fsy2k.inl, [852](#)
- \_\_FFLASFFPACK\_fflas\_fsyrk\_INL  
fflas\_fsyrk.inl, [854](#)
- \_\_FFLASFFPACK\_fflas\_fsyrk\_strassen\_INL  
fflas\_fsyrk\_strassen.inl, [855](#)
- \_\_FFLASFFPACK\_fflas\_igemm\_igemm\_INL  
igemm.inl, [860](#)
- \_\_FFLASFFPACK\_fflas\_igemm\_igemm\_kernels\_INL  
igemm\_kernels.inl, [861](#)
- \_\_FFLASFFPACK\_fflas\_igemm\_igemm\_tools\_INL  
igemm\_tools.inl, [862](#)
- \_\_FFLASFFPACK\_fflas\_pfgemm\_INL  
fflas\_pfgemm.inl, [870](#)
- \_\_FFLASFFPACK\_fflas\_pftsm\_INL  
fflas\_pftsm.inl, [871](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_pspmm\_INL  
csr\_hyb\_pspmm.inl, [896](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_pspmv\_INL  
csr\_hyb\_pspmv.inl, [896](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_spm INL  
csr\_hyb\_spm.inl, [897](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_spmv\_INL  
csr\_hyb\_spmv.inl, [897](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_utils\_INL  
csr\_hyb\_utils.inl, [898](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_pspmm\_INL  
csr\_pspmm.inl, [892](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_pspmv\_INL  
csr\_pspmv.inl, [892](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_spm INL  
csr\_spm.inl, [893](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_spmv\_INL  
csr\_spmv.inl, [894](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_pspmm\_INL  
ell\_pspmm.inl, [899](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_pspmv\_INL  
ell\_pspmv.inl, [900](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_simd\_pspmv\_INL  
ell\_simd\_pspmv.inl, [904](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_simd\_spmv\_INL  
ell\_simd\_spmv.inl, [904](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_simd\_utils\_INL  
ell\_simd\_utils.inl, [905](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_spm INL  
ell\_spm.inl, [901](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_spmv\_INL  
ell\_spmv.inl, [902](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_utils\_INL  
ell\_utils.inl, [902](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_pspmm\_INL  
hyb\_zo\_pspmm.inl, [906](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_pspmv\_INL  
hyb\_zo\_pspmv.inl, [906](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_spm INL  
hyb\_zo\_spm.inl, [907](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_spmv\_INL  
hyb\_zo\_spmv.inl, [907](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_utils\_INL  
hyb\_zo\_utils.inl, [908](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_coo\_spm INL  
coo\_spm.inl, [889](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_coo\_spmv\_INL  
coo\_spmv.inl, [890](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_coo\_utils\_INL  
coo\_utils.inl, [890](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_sell\_pspmv\_INL  
sell\_pspmv.inl, [910](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_sell\_spmv\_INL  
sell\_spmv.inl, [911](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_sell\_utils\_INL  
sell\_utils.inl, [911](#)
- \_\_FFLASFFPACK\_ffpack\_INL  
ffpack.inl, [925](#)
- \_\_FFLASFFPACK\_ffpack\_bruhatgen\_inl  
ffpack\_bruhatgen.inl, [926](#)
- \_\_FFLASFFPACK\_ffpack\_charpoly\_danilveski\_INL  
ffpack\_charpoly\_danilevski.inl, [927](#)
- \_\_FFLASFFPACK\_ffpack\_charpoly\_kgfast\_INL  
ffpack\_charpoly\_kgfast.inl, [928](#)

\_\_FFLASFFPACK\_ffpack\_charpoly\_kgfastgeneralized\_INL  
     ffpack\_charpoly\_kgfastgeneralized.inl, 928  
 \_\_FFLASFFPACK\_ffpack\_charpoly\_kglu\_INL  
     ffpack\_charpoly\_kglu.inl, 929  
 \_\_FFLASFFPACK\_ffpack\_echelon\_forms\_INL  
     ffpack\_echelonforms.inl, 931  
 \_\_FFLASFFPACK\_ffpack\_fgesv\_INL  
     ffpack\_fgesv.inl, 932  
 \_\_FFLASFFPACK\_ffpack\_fgetrs\_INL  
     ffpack\_fgetrs.inl, 932  
 \_\_FFLASFFPACK\_ffpack\_fsytrf\_INL  
     ffpack\_fsytrf.inl, 934  
 \_\_FFLASFFPACK\_ffpack\_ftrssyr2k\_INL  
     ffpack\_ftrssyr2k.inl, 935  
 \_\_FFLASFFPACK\_ffpack\_ftrstr\_INL  
     ffpack\_ftrstr.inl, 935  
 \_\_FFLASFFPACK\_ffpack\_ftrtr\_INL  
     ffpack\_ftrtr.inl, 936  
 \_\_FFLASFFPACK\_ffpack\_invert\_INL  
     ffpack\_invert.inl, 936  
 \_\_FFLASFFPACK\_ffpack\_krylovelim\_INL  
     ffpack\_krylovelim.inl, 937  
 \_\_FFLASFFPACK\_ffpack\_ludivine\_INL  
     ffpack\_ludivine.inl, 937  
 \_\_FFLASFFPACK\_ffpack\_minpoly\_INL  
     ffpack\_minpoly.inl, 939  
 \_\_FFLASFFPACK\_ffpack\_permutation\_INL  
     ffpack\_permutation.inl, 941  
 \_\_FFLASFFPACK\_ffpack\_pluq\_INL  
     ffpack\_pluq.inl, 942  
 \_\_FFLASFFPACK\_ffpack\_ppluq\_INL  
     ffpack\_ppluq.inl, 943  
 \_\_FFLASFFPACK\_ffpack\_rank\_profiles\_INL  
     ffpack\_rankprofiles.inl, 944  
 \_\_FFLASFFPACK\_ffgemm\_INL  
     fflas\_fgemm.inl, 835  
 \_\_FFLASFFPACK\_ffgemm\_bini\_INL  
     schedule\_bini.inl, 839  
 \_\_FFLASFFPACK\_ffgemm\_winograd\_INL  
     schedule\_winograd.inl, 840  
 \_\_FFLASFFPACK\_ffgemm\_winograd\_acc\_INL  
     schedule\_winograd\_acc.inl, 841  
 \_\_FFLASFFPACK\_ffgemm\_winograd\_acc\_ip\_INL  
     schedule\_winograd\_acc\_ip.inl, 841  
 \_\_FFLASFFPACK\_ffgemm\_winograd\_ip\_INL  
     schedule\_winograd\_ip.inl, 842  
 \_\_FFLASFFPACK\_ffgemv\_INL  
     fflas\_ffgemv.inl, 843  
 \_\_FFLASFFPACK\_ffgemv\_mp\_INL  
     fflas\_ffgemv\_mp.inl, 844  
 \_\_FFLASFFPACK\_ffger\_INL  
     fflas\_ffger.inl, 845  
 \_\_FFLASFFPACK\_field\_rns\_INL  
     rns.inl, 951  
 \_\_FFLASFFPACK\_field\_rns\_double\_INL  
     rns-double.inl, 949  
 \_\_FFLASFFPACK\_field\_rns\_double\_recint\_INL  
     rns-double-recint.inl, 948  
 \_\_FFLASFFPACK\_freivalds\_INL  
     fflas\_freivalds.inl, 849  
 \_\_FFLASFFPACK\_fscal\_INL  
     fflas\_fscal.inl, 851  
 \_\_FFLASFFPACK\_fscal\_mp\_INL  
     fflas\_fscal\_mp.inl, 852  
 \_\_FFLASFFPACK\_ftrmm\_INL  
     fflas\_ftrmm.inl, 856  
 \_\_FFLASFFPACK\_ftrsm\_INL  
     fflas\_ftrsm.inl, 856  
 \_\_FFLASFFPACK\_ftrsv\_INL  
     fflas\_ftrsv.inl, 858  
 \_\_FFLASFFPACK\_simd512\_INL  
     simd512.inl, 879  
 \_\_FFLASFFPACK\_simd512\_double\_INL  
     simd512\_double.inl, 880  
 \_\_FFLASFFPACK\_simd512\_float\_INL  
     simd512\_float.inl, 880  
 \_\_FFLASFFPACK\_simd512\_int32\_INL  
     simd512\_int32.inl, 880  
 \_\_FFLAS\_L1\_INST\_C  
     fflas\_L1\_inst.C, 963  
 \_\_FFLAS\_L2\_INST\_C  
     fflas\_L2\_inst.C, 967  
 \_\_FFLAS\_L3\_INST\_C  
     fflas\_L3\_inst.C, 970  
 \_\_FFLAS\_\_TRSM\_READONLY  
     fflas\_L3\_inst\_implement.inl, 973  
     fflas\_level3.inl, 870  
     ffpack\_ppluq.inl, 943  
 \_\_FFPACK\_FSYTRF\_BC\_CROUT  
     benchmark-fsytrf.C, 796  
 \_\_FFPACK\_INST\_C  
     ffpack\_inst.C, 1025  
 \_\_FFPACK\_charpoly\_mp\_INL  
     ffpack\_charpoly\_mp.inl, 929  
 \_\_FFPACK\_det\_mp\_INL  
     ffpack\_det\_mp.inl, 930  
 \_\_FFPACK\_ffgemm\_classical\_INL  
     ffgemm\_classical\_mp.inl, 837  
 \_\_FFPACK\_fger\_mp\_INL  
     fflas\_fger\_mp.inl, 846  
 \_\_FFPACK\_ftrsm\_mp\_INL  
     fflas\_ftrsm\_mp.inl, 857  
 \_\_FFPACK\_ludivine\_mp\_INL  
     ffpack\_ludivine\_mp.inl, 938  
 \_\_FFPACK\_pluq\_mp\_INL  
     ffpack\_pluq\_mp.inl, 942  
 \_\_LUDIVINE\_CUTOFF  
     test-lu.C, 1089  
 \_\_has\_builtin  
     bit\_manipulation.h, 1042  
 \_\_alloc  
     rns\_double\_elt, 532  
     rns\_double\_elt\_cstptr, 535  
     rns\_double\_elt\_ptr, 538  
 \_\_basis  
     rns\_double, 530

- rns\_double\_extended, [541](#)
- \_basisMax
  - rns\_double, [530](#)
  - rns\_double\_extended, [541](#)
- \_coo
  - SpMat< Field, flag >, [756](#)
- \_coo16
  - CooMat< Field >, [435](#)
- \_coo16\_zo
  - CooMat< Field >, [435](#)
- \_coo32
  - CooMat< Field >, [435](#)
- \_coo32\_zo
  - CooMat< Field >, [435](#)
- \_coo64
  - CooMat< Field >, [435](#)
- \_coo64\_zo
  - CooMat< Field >, [435](#)
- \_crt\_in
  - rns\_double, [531](#)
  - rns\_double\_extended, [541](#)
- \_crt\_out
  - rns\_double, [531](#)
  - rns\_double\_extended, [541](#)
- \_csr
  - SpMat< Field, flag >, [756](#)
- \_csr16
  - CsrMat< Field >, [437](#)
- \_csr16\_zo
  - CsrMat< Field >, [437](#)
- \_csr32
  - CsrMat< Field >, [437](#)
- \_csr32\_zo
  - CsrMat< Field >, [437](#)
- \_csr64
  - CsrMat< Field >, [437](#)
- \_csr64\_zo
  - CsrMat< Field >, [437](#)
- \_ell
  - SpMat< Field, flag >, [756](#)
- \_ell16
  - EllMat< Field >, [443](#)
- \_ell16\_zo
  - EllMat< Field >, [443](#)
- \_ell32
  - EllMat< Field >, [443](#)
- \_ell32\_zo
  - EllMat< Field >, [444](#)
- \_ell64
  - EllMat< Field >, [443](#)
- \_ell64\_zo
  - EllMat< Field >, [444](#)
- \_errorStream
  - Failure, [445](#)
- \_field\_rns
  - rns\_double, [530](#)
  - rns\_double\_extended, [541](#)
- \_iM\_modp\_rns
  - RNSIntegerMod< RNS >, [551](#)
- \_ibeg
  - ForStrategy2D< blocksize\_t, Cut, Param >, [465](#)
- \_iend
  - ForStrategy2D< blocksize\_t, Cut, Param >, [465](#)
- \_invbasis
  - rns\_double, [530](#)
  - rns\_double\_extended, [541](#)
- \_jbeg
  - ForStrategy2D< blocksize\_t, Cut, Param >, [465](#)
- \_jend
  - ForStrategy2D< blocksize\_t, Cut, Param >, [465](#)
- \_ldm
  - rns\_double, [531](#)
  - rns\_double\_extended, [541](#)
- \_mi\_sum
  - rns\_double, [531](#)
- \_mm
  - Test< Elt >, [764](#)
- \_negbasis
  - rns\_double, [530](#)
  - rns\_double\_extended, [541](#)
- \_nn
  - Test< Elt >, [764](#)
- \_p
  - RNSIntegerMod< RNS >, [551](#)
- \_pbits
  - rns\_double, [531](#)
  - rns\_double\_extended, [541](#)
- \_ptr
  - rns\_double\_elt, [532](#)
  - rns\_double\_elt\_cstptr, [535](#)
  - rns\_double\_elt\_ptr, [538](#)
- \_rns
  - RNSInteger< RNS >, [545](#)
  - RNSIntegerMod< RNS >, [551](#)
- \_simd512\_int64\_INL
  - simd512\_int64.inl, [881](#)
- \_size
  - rns\_double, [531](#)
  - rns\_double\_extended, [541](#)
- \_stride
  - rns\_double\_elt, [532](#)
  - rns\_double\_elt\_cstptr, [535](#)
  - rns\_double\_elt\_ptr, [538](#)
- \_zero
  - ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >, [560](#)
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, [563](#)
- ~CheckerImplem\_Det
  - CheckerImplem\_Det< Field >, [415](#)
- ~CheckerImplem\_PLUQ
  - CheckerImplem\_PLUQ< Field >, [420](#)
- ~CheckerImplem\_charpoly
  - CheckerImplem\_charpoly< Field, Polynomial >, [413](#)



- CheckerImplem\_charpoly< Givaro::ZRing< Givaro::Integer >, Polynomial >, 414
- ~CheckerImplem\_fgemm
  - CheckerImplem\_fgemm< Field >, 417
- ~CheckerImplem\_ftsm
  - CheckerImplem\_ftsm< Field >, 418
- ~CheckerImplem\_invert
  - CheckerImplem\_invert< Field >, 419
- ~rns\_double\_elt
  - rns\_double\_elt, 532
- 101-fgemm.C, 1104
  - main, 1105
- 2x2-fgemm.C, 1105
  - main, 1105
- 2x2-ftsv.C, 1105
  - main, 1105
- 2x2-pluq.C, 1106
  - main, 1106
- add
  - FFLAS::vectorised, 281
  - FieldSimd< \_Field >, 448
  - RNSIntegerMod< RNS >, 549
  - ScalFunctions< Element >, 555
  - Simd128\_impl< true, true, false, 2 >, 572
  - Simd128\_impl< true, true, false, 4 >, 581
  - Simd128\_impl< true, true, false, 8 >, 592
  - Simd128\_impl< true, true, true, 2 >, 599
  - Simd128\_impl< true, true, true, 4 >, 608
  - Simd128\_impl< true, true, true, 8 >, 617
  - Simd256\_impl< true, false, true, 8 >, 628
  - Simd256\_impl< true, true, false, 2 >, 639
  - Simd256\_impl< true, true, false, 4 >, 655
  - Simd256\_impl< true, true, false, 8 >, 667
  - Simd256\_impl< true, true, true, 2 >, 675
  - Simd256\_impl< true, true, true, 4 >, 685, 691
  - Simd256\_impl< true, true, true, 8 >, 701
  - Simd512\_impl< true, false, true, 8 >, 710
  - Simd512\_impl< true, true, false, 8 >, 721
  - Simd512\_impl< true, true, true, 8 >, 729
- add\_r
  - FieldSimd< \_Field >, 449
- addin
  - FieldSimd< \_Field >, 449
  - ScalFunctions< Element >, 555
  - Simd128\_impl< true, true, false, 2 >, 572
  - Simd128\_impl< true, true, false, 4 >, 582
  - Simd128\_impl< true, true, false, 8 >, 592
  - Simd128\_impl< true, true, true, 2 >, 600
  - Simd128\_impl< true, true, true, 4 >, 608
  - Simd128\_impl< true, true, true, 8 >, 618
  - Simd256\_impl< true, false, true, 8 >, 628
  - Simd256\_impl< true, true, false, 2 >, 639
  - Simd256\_impl< true, true, false, 4 >, 655, 656
  - Simd256\_impl< true, true, false, 8 >, 667
  - Simd256\_impl< true, true, true, 2 >, 675
  - Simd256\_impl< true, true, true, 4 >, 685, 691
  - Simd256\_impl< true, true, true, 8 >, 701
  - Simd512\_impl< true, false, true, 8 >, 710
- Simd512\_impl< true, true, false, 8 >, 721
- Simd512\_impl< true, true, true, 8 >, 729
- addin\_r
  - FieldSimd< \_Field >, 449
- addp
  - FFLAS::vectorised, 280
- AlgoChooser< ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq >, 401
  - value, 401
- AlgoChooser< ModeT, ParSeq >, 401
  - value, 401
- align-allocator.h, 1040
- alignable
  - FFLAS, 188
- alignable< Givaro::Integer \* >
  - FFLAS, 188
- aligned\_allocator
  - NoSimd< T >, 521
  - Simd128\_impl< true, true, false, 2 >, 566
  - Simd128\_impl< true, true, false, 4 >, 576
  - Simd128\_impl< true, true, false, 8 >, 586
  - Simd128\_impl< true, true, true, 2 >, 596
  - Simd128\_impl< true, true, true, 4 >, 605
  - Simd128\_impl< true, true, true, 8 >, 615
  - Simd256\_impl< true, false, true, 8 >, 625
  - Simd256\_impl< true, true, false, 2 >, 633
  - Simd256\_impl< true, true, false, 4 >, 645
  - Simd256\_impl< true, true, false, 8 >, 662
  - Simd256\_impl< true, true, true, 2 >, 671
  - Simd256\_impl< true, true, true, 4 >, 682
  - Simd256\_impl< true, true, true, 8 >, 698
  - Simd512\_impl< true, false, true, 8 >, 707
  - Simd512\_impl< true, true, false, 8 >, 715
  - Simd512\_impl< true, true, true, 8 >, 726
- aligned\_vector
  - NoSimd< T >, 521
  - Simd128\_impl< true, true, false, 2 >, 566
  - Simd128\_impl< true, true, false, 4 >, 576
  - Simd128\_impl< true, true, false, 8 >, 586
  - Simd128\_impl< true, true, true, 2 >, 597
  - Simd128\_impl< true, true, true, 4 >, 606
  - Simd128\_impl< true, true, true, 8 >, 615
  - Simd256\_impl< true, false, true, 8 >, 625
  - Simd256\_impl< true, true, false, 2 >, 633
  - Simd256\_impl< true, true, false, 4 >, 645
  - Simd256\_impl< true, true, false, 8 >, 662
  - Simd256\_impl< true, true, true, 2 >, 671
  - Simd256\_impl< true, true, true, 4 >, 682
  - Simd256\_impl< true, true, true, 8 >, 698
  - Simd512\_impl< true, false, true, 8 >, 707
  - Simd512\_impl< true, true, false, 8 >, 715
  - Simd512\_impl< true, true, true, 8 >, 726
- alignment
  - FieldSimd< \_Field >, 452
  - NoSimd< T >, 521
  - Simd128\_impl< true, true, false, 2 >, 574
  - Simd128\_impl< true, true, false, 4 >, 584



- Simd128\_impl< true, true, false, 8 >, [594](#)
- Simd128\_impl< true, true, true, 2 >, [603](#)
- Simd128\_impl< true, true, true, 4 >, [612](#)
- Simd128\_impl< true, true, true, 8 >, [622](#)
- Simd256\_impl< true, false, true, 8 >, [631](#)
- Simd256\_impl< true, true, false, 2 >, [641](#)
- Simd256\_impl< true, true, false, 4 >, [659](#)
- Simd256\_impl< true, true, false, 8 >, [669](#)
- Simd256\_impl< true, true, true, 2 >, [678](#)
- Simd256\_impl< true, true, true, 4 >, [695](#)
- Simd256\_impl< true, true, true, 8 >, [704](#)
- Simd512\_impl< true, false, true, 8 >, [712](#)
- Simd512\_impl< true, true, false, 8 >, [723](#)
- Simd512\_impl< true, true, true, 8 >, [733](#)
- ALL< false, v... >, [402](#)
  - value, [402](#)
- ALL< true, v... >, [402](#)
  - value, [402](#)
- ALL< v >, [401](#)
- ALL<>, [402](#)
  - value, [402](#)
- Amax
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [502](#)
- Amin
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [502](#)
- applyP
  - FFPACK, [304](#), [305](#), [365](#)
- applyP\_block
  - FFPACK, [357](#)
- applyP\_modular\_double
  - ffpack.C, [990](#)
  - ffpack\_c.h, [1011](#)
- ArbitraryPreIntTag, [402](#)
- Architecture of the library., [2](#)
- areEqual
  - RNSIntegerMod< RNS >, [550](#)
- AreEqual< X, X >, [403](#)
  - value, [403](#)
- AreEqual< X, Y >, [403](#)
  - value, [403](#)
- args-parser.h, [1040](#)
  - ArgumentType, [1041](#)
  - END\_OF\_ARGUMENTS, [1041](#)
  - findArgument, [1042](#)
  - getListArgs, [1042](#)
  - printHelpMessage, [1042](#)
  - TYPE\_BOOL, [1041](#)
  - TYPE\_DOUBLE, [1041](#)
  - TYPE\_INT, [1041](#)
  - TYPE\_INTEGER, [1041](#)
  - type\_integer, [1041](#)
  - TYPE\_INTLIST, [1041](#)
  - TYPE\_LONGLONG, [1041](#)
  - TYPE\_NONE, [1041](#)
  - TYPE\_STR, [1041](#)
  - TYPE\_UINT64, [1041](#)
- Argument, [403](#)
  - c, [403](#)
  - data, [404](#)
  - example, [403](#)
  - helpString, [403](#)
  - type, [404](#)
- ArgumentType
  - args-parser.h, [1041](#)
- ArithProg
  - FFPACK::Protected, [395](#)
- arithprog.C, [773](#)
  - main, [773](#)
  - TTimer, [773](#)
- assign
  - RNSInteger< RNS >, [545](#)
  - RNSIntegerMod< RNS >, [549](#)
- associatedDelayedField< const FFPACK::RNSIntegerMod< RNS > >, [404](#)
  - field, [404](#)
  - type, [404](#)
- associatedDelayedField< const Givaro::Modular< T, X > >, [405](#)
  - field, [405](#)
  - type, [405](#)
- associatedDelayedField< const Givaro::ModularBalanced< T > >, [405](#)
  - field, [405](#)
  - type, [405](#)
- associatedDelayedField< const Givaro::ZRing< T > >, [406](#)
  - field, [406](#)
  - type, [406](#)
- associatedDelayedField< Field >, [404](#)
  - field, [404](#)
  - type, [404](#)
- assume\_aligned
  - fflas\_sparse.h, [885](#)
- AtlasConj
  - config-blas.h, [812](#)
- Aunfit
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [501](#)
- aut
  - HelperFlag, [474](#)
- Auto, [406](#)
- averageCol
  - StatsMatrix, [758](#)
- averageColDifference
  - StatsMatrix, [758](#)
- averageRow
  - StatsMatrix, [758](#)
- averageRowDifference
  - StatsMatrix, [758](#)
- axpy
  - FieldSimd< \_Field >, [451](#)
- axpy\_r
  - FieldSimd< \_Field >, [451](#)
- axpyin

- FieldSimd< \_Field >, 451
- RNSIntegerMod< RNS >, 550
- axpyin\_r
  - FieldSimd< \_Field >, 451
- axpyp
  - FFLAS::vectorised, 281
  - FFLAS::vectorised::unswitch, 285
- balanced
  - FieldTraits< FFPACK::RNSInteger< T > >, 453
  - FieldTraits< FFPACK::RNSIntegerMod< T > >, 454
  - FieldTraits< Field >, 453
  - FieldTraits< Givaro::Modular< Element > >, 454
  - FieldTraits< Givaro::ModularBalanced< Element > >, 455
  - FieldTraits< Givaro::ZRing< double > >, 455
  - FieldTraits< Givaro::ZRing< float > >, 456
  - FieldTraits< Givaro::ZRing< Givaro::Integer > >, 456
  - FieldTraits< Givaro::ZRing< int16\_t > >, 457
  - FieldTraits< Givaro::ZRing< int32\_t > >, 457
  - FieldTraits< Givaro::ZRing< int64\_t > >, 458
  - FieldTraits< Givaro::ZRing< Reclnt::ruint< K > >, 458
  - FieldTraits< Givaro::ZRing< uint16\_t > >, 459
  - FieldTraits< Givaro::ZRing< uint32\_t > >, 459
  - FieldTraits< Givaro::ZRing< uint64\_t > >, 459
  - winograd.C, 779
- BARRIER
  - parallel.h, 1033
- BASECASE\_K
  - test-lu.C, 1089
- BaseTimer
  - FFLAS, 74
- BasisElement
  - rns\_double, 528
  - rns\_double\_extended, 539
  - RNSInteger< RNS >, 543
  - RNSIntegerMod< RNS >, 547
- begin
  - ForStrategy1D< blocksize\_t, Cut, Param >, 462
  - Info, 479, 480
- BEGIN\_PARALLEL\_MAIN
  - parallel.h, 1034
- Bench
  - Bench< Elt >, 408
- Bench< Elt >, 406
  - Bench, 408
  - cardinality, 408
  - doBenchs, 408
  - Elt\_ptr, 407
  - enable\_if\_no\_simd\_t, 407
  - enable\_if\_simd128\_t, 407
  - enable\_if\_simd256\_t, 407
  - enable\_if\_simd512\_t, 407
  - enable\_if\_t, 407
  - F, 408
  - Field, 407
  - inplace, 409
  - is\_same\_element, 407
  - iters, 409
  - m, 408
  - n, 408
  - Residu, 407
  - run, 408
- benchmark-charpoly-mp.C, 779
  - \_\_FFLASFFPACK\_FORCE\_SEQ, 779
  - main, 779
- benchmark-charpoly.C, 779
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, 780
  - main, 780
  - run\_with\_field, 780
- benchmark-checkers.C, 780
  - \_MAX\_SIZE\_MATRICES, 781
  - \_NR\_TESTS, 781
  - CUBE, 781
  - ENABLE\_ALL\_CHECKINGS, 781
  - main, 781
- benchmark-dgemm.C, 781
  - CBLAS\_GEMM, 782
  - Floats, 782
  - main, 782
  - TTimer, 782
- benchmark-dgetrf.C, 782
  - \_\_FFLASFFPACK\_HAVE\_DGETRF, 782
  - main, 783
- benchmark-dgetri.C, 783
  - main, 783
- benchmark-dsytrf.C, 783
  - EFFGFF, 783
  - main, 784
- benchmark-dtrsm.C, 784
  - main, 784
- benchmark-dtrtri.C, 784
  - \_\_FFLASFFPACK\_HAVE\_DTRTRI, 785
  - main, 785
- benchmark-fadd-lvl2.C, 785
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, 785
  - main, 785
- benchmark-fdot.C, 785
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, 786
  - main, 786
  - run\_with\_field, 786
- benchmark-fgemm-mp.C, 786
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, 787
  - main, 787
  - MG\_DEFAULT, 787
  - STD\_RECINT\_SIZE, 787
  - tmain, 787
- benchmark-fgemm-rns.C, 787
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, 788

- ConstElement\_ptr, [788](#)
- Element\_ptr, [788](#)
- Field, [788](#)
- GRAIN, [788](#)
- main, [789](#)
- PSeq, [789](#)
- RNS, [788](#)
- THREADS, [788](#)
- THREED, [788](#)
- THREEDA, [788](#)
- THREEDIP, [788](#)
- TWOD, [788](#)
- TWODA, [788](#)
- benchmark-fgemm.C, [789](#)
  - CLASSIC\_HYBRID, [789](#)
  - main, [789](#)
- benchmark-fgemv-mp.C, [789](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [790](#)
  - main, [790](#)
  - MG\_DEFAULT, [790](#)
  - STD\_RECINT\_SIZE, [790](#)
  - tmain, [790](#)
  - write\_matrix, [790](#)
- benchmark-fgemv.C, [791](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [791](#)
  - benchmark\_disp, [793](#)
  - benchmark\_in\_Field, [793](#)
  - benchmark\_with\_field, [793](#)
  - benchmark\_with\_timer, [792](#)
  - check\_result, [792](#)
  - fill\_value, [792](#)
  - genData, [792](#)
  - main, [794](#)
- benchmark-fgesv.C, [794](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [794](#)
  - main, [794](#)
- benchmark-fsyr2k.C, [794](#)
  - main, [795](#)
- benchmark-fsyrk.C, [795](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [795](#)
  - main, [795](#)
- benchmark-fsytrf.C, [795](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [796](#)
  - \_\_FFPACK\_FSYTRF\_BC\_CROUT, [796](#)
  - CUBE, [796](#)
  - main, [796](#)
- benchmark-ftdsm-mp.C, [796](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [796](#)
  - main, [797](#)
- benchmark-ftdsm.C, [797](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [797](#)
  - main, [797](#)
- benchmark-ftsv.C, [797](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [798](#)
  - main, [798](#)
- benchmark-fttri.C, [798](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [798](#)
  - CUBE, [798](#)
  - main, [798](#)
- benchmark-inverse.C, [798](#)
  - CUBE, [799](#)
  - main, [799](#)
- benchmark-lqup-mp.C, [799](#)
  - main, [799](#)
- benchmark-lqup.C, [800](#)
  - CUBE, [800](#)
  - main, [800](#)
- benchmark-pluq.C, [800](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [801](#)
  - CUBE, [801](#)
  - Field, [801](#)
  - main, [801](#)
  - Rec\_Initialize, [801](#)
  - verification\_PLUQ, [801](#)
- benchmark-quasiseq.C, [801](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [802](#)
  - main, [802](#)
  - run\_with\_field, [802](#)
- benchmark-storage-transpose.C, [802](#)
  - main, [803](#)
- benchmark-wino.C, [803](#)
  - CUBE, [803](#)
  - launch\_wino, [803](#)
  - main, [803](#)
- benchmark\_disp
  - benchmark-fgemv.C, [793](#)
- benchmark\_in\_Field
  - benchmark-fgemv.C, [793](#)
- benchmark\_with\_field
  - benchmark-fgemv.C, [793](#)
- benchmark\_with\_timer
  - benchmark-fgemv.C, [792](#)
- Bibliography, [7](#)
- Bini, [409](#)
- FFLAS::BLAS3, [192](#)
- bit\_manipulation.h, [1042](#)
  - \_\_has\_builtin, [1042](#)
  - clz, [1043](#)
  - ctz, [1043](#)
- bitsize
  - FFLAS, [140](#)
- bitsize< Givaro::ZRing< Givaro::Integer > >
  - FFLAS, [140](#)
- blas\_enum
  - config-blas.h, [811](#)

- blend
  - ScalFunctions< Element >, [558](#)
  - Simd128\_impl< true, true, false, 2 >, [572](#)
  - Simd128\_impl< true, true, false, 4 >, [581](#)
  - Simd128\_impl< true, true, false, 8 >, [591](#)
  - Simd128\_impl< true, true, true, 2 >, [599](#)
  - Simd128\_impl< true, true, true, 4 >, [608](#)
  - Simd128\_impl< true, true, true, 8 >, [617](#)
  - Simd256\_impl< true, false, true, 8 >, [628](#)
  - Simd256\_impl< true, true, false, 2 >, [639](#)
  - Simd256\_impl< true, true, false, 4 >, [655](#)
  - Simd256\_impl< true, true, false, 8 >, [667](#)
  - Simd256\_impl< true, true, true, 2 >, [675](#)
  - Simd256\_impl< true, true, true, 4 >, [685](#), [691](#)
  - Simd256\_impl< true, true, true, 8 >, [700](#)
  - Simd512\_impl< true, false, true, 8 >, [709](#)
  - Simd512\_impl< true, true, false, 8 >, [721](#)
  - Simd512\_impl< true, true, true, 8 >, [729](#)
- blendv
  - ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >, [559](#)
  - Simd256\_impl< true, false, true, 8 >, [628](#)
  - Simd512\_impl< true, false, true, 8 >, [710](#)
- Block, [409](#)
- BlockCuts
  - FFLAS, [179](#), [181](#)
- BlockCuts< CuttingStrategy::Block, StrategyParameter::Fixed >
  - FFLAS, [180](#)
- BlockCuts< CuttingStrategy::Block, StrategyParameter::Grain >
  - FFLAS, [179](#)
- BlockCuts< CuttingStrategy::Block, StrategyParameter::Threads >
  - FFLAS, [181](#)
- BlockCuts< CuttingStrategy::Column, StrategyParameter::Fixed >
  - FFLAS, [180](#)
- BlockCuts< CuttingStrategy::Column, StrategyParameter::Grain >
  - FFLAS, [180](#)
- BlockCuts< CuttingStrategy::Column, StrategyParameter::Threads >
  - FFLAS, [180](#)
- BlockCuts< CuttingStrategy::Row, StrategyParameter::Fixed >
  - FFLAS, [179](#)
- BlockCuts< CuttingStrategy::Row, StrategyParameter::Grain >
  - FFLAS, [179](#)
- BlockCuts< CuttingStrategy::Row, StrategyParameter::Threads >
  - FFLAS, [180](#)
- BlockCuts< CuttingStrategy::Single, StrategyParameter::Threads >
  - FFLAS, [179](#)
- blockcuts.inl, [1030](#)
- \_\_FFLASFFPACK\_MINBLOCKCUTS, [1031](#)
- \_\_FFLASFFPACK\_fflas\_blockcuts\_INL, [1031](#)
- blockindex
  - ForStrategy1D< blocksize\_t, Cut, Param >, [462](#)
  - ForStrategy2D< blocksize\_t, Cut, Param >, [465](#)
- BlockingFactor
  - FFLAS::details, [206](#)
- BLOCKS
  - ForStrategy2D< blocksize\_t, Cut, Param >, [466](#)
- BlockTransposeSIMD< Field, Simd, >, [409](#)
  - info, [409](#)
  - size, [409](#)
  - transpose, [410](#)
- Bmax
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, [502](#)
- Bmin
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, [502](#)
- Bruhat2EchelonPermutation
  - FFPACK, [345](#)
- Bug List, [3](#)
- build
  - ForStrategy1D< blocksize\_t, Cut, Param >, [461](#)
- buildMatrix
  - FFPACK, [351](#)
- Bunfit
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, [501](#)
- c
  - Argument, [403](#)
- callLUdivine\_small< double >, [411](#)
  - operator(), [411](#)
- callLUdivine\_small< Element >, [411](#)
  - operator(), [411](#)
- callLUdivine\_small< float >, [412](#)
  - operator(), [412](#)
- cardinality
  - Bench< Elt >, [408](#)
  - RNSInteger< RNS >, [544](#)
  - RNSIntegerMod< RNS >, [548](#)
  - Test< Elt >, [763](#)
- cast.h, [1043](#)
- category
  - FieldTraits< FFPACK::RNSInteger< T > >, [453](#)
  - FieldTraits< FFPACK::RNSIntegerMod< T > >, [454](#)
  - FieldTraits< Field >, [453](#)
  - FieldTraits< Givaro::Modular< Element > >, [454](#)
  - FieldTraits< Givaro::ModularBalanced< Element > >, [455](#)
  - FieldTraits< Givaro::ZRing< double > >, [455](#)
  - FieldTraits< Givaro::ZRing< float > >, [456](#)
  - FieldTraits< Givaro::ZRing< Givaro::Integer > >, [456](#)
  - FieldTraits< Givaro::ZRing< int16\_t > >, [457](#)
  - FieldTraits< Givaro::ZRing< int32\_t > >, [457](#)
  - FieldTraits< Givaro::ZRing< int64\_t > >, [457](#)

- FieldTraits< Givaro::ZRing< ReclInt::ruint< K > >  
>, [458](#)
- FieldTraits< Givaro::ZRing< uint16\_t > >, [458](#)
- FieldTraits< Givaro::ZRing< uint32\_t > >, [459](#)
- FieldTraits< Givaro::ZRing< uint64\_t > >, [459](#)
- cblas.C, [1050](#)
  - \_\_FFLASFFPACK\_CONFIGURATION, [1051](#)
  - \_\_FFLASFFPACK\_HAVE\_CBLAS, [1051](#)
  - main, [1051](#)
- CBLAS\_DIAG
  - config-blas.h, [812](#)
- cblas\_dsyrk
  - config-blas.h, [816](#)
- CBLAS\_ENUM\_DEFINED\_H
  - config-blas.h, [811](#)
- CBLAS\_EXTERNALS
  - config-blas.h, [811](#)
- CBLAS\_GEMM
  - benchmark-dgemm.C, [782](#)
- cblas\_imptrsm
  - FFLAS, [127](#)
- CBLAS\_INT
  - config-blas.h, [811](#)
- CBLAS\_ORDER
  - config-blas.h, [811](#)
- CBLAS\_SIDE
  - config-blas.h, [812](#)
- CBLAS\_TRANSPOSE
  - config-blas.h, [811](#)
- CBLAS\_UPLO
  - config-blas.h, [812](#)
- CblasColMajor
  - config-blas.h, [811](#)
- CblasConjTrans
  - config-blas.h, [812](#)
- CblasLeft
  - config-blas.h, [812](#)
- CblasLower
  - config-blas.h, [812](#)
- CblasNonUnit
  - config-blas.h, [812](#)
- CblasNoTrans
  - config-blas.h, [812](#)
- CblasRight
  - config-blas.h, [812](#)
- CblasRowMajor
  - config-blas.h, [811](#)
- CblasTrans
  - config-blas.h, [812](#)
- CblasUnit
  - config-blas.h, [812](#)
- CblasUpper
  - config-blas.h, [812](#)
- ceil
  - ScalFunctionsBase< Element, typename enable\_if<  
is\_floating\_point< Element >::value >::type  
>, [559](#)
  - Simd256\_impl< true, false, true, 8 >, [630](#)
  - Simd512\_impl< true, false, true, 8 >, [712](#)
- changeBS
  - ForStrategy1D< blocksize\_t, Cut, Param >, [463](#)
- changeCBS
  - ForStrategy2D< blocksize\_t, Cut, Param >, [466](#)
- changeRBS
  - ForStrategy2D< blocksize\_t, Cut, Param >, [466](#)
- characteristic
  - RNSInteger< RNS >, [544](#)
  - RNSIntegerMod< RNS >, [548](#)
- CharPoly
  - FFPACK, [323](#), [324](#), [351](#), [369](#), [370](#)
- charpoly.C, [773](#), [774](#)
  - CUBE, [774](#)
  - GFOPS, [774](#)
  - main, [774](#)
  - TTimer, [774](#)
- CharpolyFailed, [412](#)
- check
  - Checker\_Empty< Field >, [413](#)
  - CheckerImplem\_charpoly< Field, Polynomial >, [414](#)
  - CheckerImplem\_charpoly< Givaro::ZRing< Givaro::Integer >, Polynomial >, [415](#)
  - CheckerImplem\_Det< Field >, [415](#)
  - CheckerImplem\_fgemm< Field >, [418](#)
  - CheckerImplem\_ftsm< Field >, [419](#)
  - CheckerImplem\_invert< Field >, [420](#)
  - CheckerImplem\_PLUQ< Field >, [420](#)
- check1
  - regression-check.C, [1053](#)
- check2
  - regression-check.C, [1053](#)
- check3
  - regression-check.C, [1054](#)
- check4
  - regression-check.C, [1054](#)
- check\_computeS1S2
  - test-fsyrc.C, [1076](#)
- CHECK\_DEPENDENCIES
  - parallel.h, [1033](#)
- check\_eq
  - test-simd.C, [1102](#)
- check\_fdot
  - test-fdot.C, [1062](#)
- check\_fger
  - test-fger.C, [1069](#)
- check\_fsyr2k
  - test-fsyr2k.C, [1074](#)
- check\_fsyrc
  - test-fsyrc.C, [1075](#)
- check\_fsyrc\_bkdiag
  - test-fsyrc.C, [1076](#)
- check\_fsyrc\_diag
  - test-fsyrc.C, [1075](#)
- check\_ftmm
  - test-ftmm.C, [1078](#)
- check\_ftmmv

- test-ftrmv.C, [1080](#)
- check\_ftrsm
  - test-ftrsm.C, [1081](#)
- check\_ftrssyr2k
  - test-ftrssyr2k.C, [1082](#)
- check\_ftrstr
  - test-ftrstr.C, [1084](#)
- check\_ftrsv
  - test-ftrsv.C, [1085](#)
- check\_ftrtri
  - test-ftrtri.C, [1086](#)
- check\_minpoly
  - test-minpoly.C, [1093](#)
- check\_MM
  - test-fgemm.C, [1065](#)
- check\_MV
  - test-fgemv.C, [1067](#)
- check\_result
  - benchmark-fgemv.C, [792](#)
- check\_solve
  - test-solve.C, [1103](#)
- checkA
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [501](#)
- checkB
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [501](#)
- CHECKER, [41](#)
- Checker\_charpoly
  - FFPACK, [303](#)
- checker\_charpoly.inl, [805](#)
  - \_\_FFLASFFPACK\_checker\_charpoly\_INL, [806](#)
- Checker\_Det
  - FFPACK, [303](#)
- checker\_det.inl, [806](#)
  - \_\_FFLASFFPACK\_checker\_det\_INL, [806](#)
- Checker\_Empty
  - Checker\_Empty< Field >, [413](#)
- Checker\_Empty< Field >, [412](#)
  - check, [413](#)
  - Checker\_Empty, [413](#)
- checker\_empty.h, [806](#)
- Checker\_fgemm
  - FFLAS, [72](#)
- checker\_fgemm.inl, [806](#)
  - \_\_FFLASFFPACK\_checker\_fgemm\_INL, [807](#)
- Checker\_ftrsm
  - FFLAS, [72](#)
- checker\_ftrsm.inl, [807](#)
  - \_\_FFLASFFPACK\_checker\_ftrsm\_INL, [807](#)
- Checker\_invert
  - FFPACK, [303](#)
- checker\_invert.inl, [807](#)
  - \_\_FFLASFFPACK\_checker\_invert\_INL, [807](#)
- Checker\_PLUQ
  - FFPACK, [303](#)
- checker\_pluq.inl, [807](#)
  - \_\_FFLASFFPACK\_checker\_pluq\_INL, [808](#)
- CheckerImplem\_charpoly
  - CheckerImplem\_charpoly< Field, Polynomial >, [413](#)
  - CheckerImplem\_charpoly< Givaro::ZRing< Givaro::Integer >, Polynomial >, [414](#)
- CheckerImplem\_charpoly< Field, Polynomial >, [413](#)
  - ~CheckerImplem\_charpoly, [413](#)
  - check, [414](#)
  - CheckerImplem\_charpoly, [413](#)
- CheckerImplem\_charpoly< Givaro::ZRing< Givaro::Integer >, Polynomial >, [414](#)
  - ~CheckerImplem\_charpoly, [414](#)
  - check, [415](#)
  - CheckerImplem\_charpoly, [414](#)
  - Ring, [414](#)
- CheckerImplem\_Det
  - CheckerImplem\_Det< Field >, [415](#)
- CheckerImplem\_Det< Field >, [415](#)
  - ~CheckerImplem\_Det, [415](#)
  - check, [415](#)
  - CheckerImplem\_Det, [415](#)
- CheckerImplem\_fgemm
  - CheckerImplem\_fgemm< Field >, [417](#)
- CheckerImplem\_fgemm< Field >, [417](#)
  - ~CheckerImplem\_fgemm, [417](#)
  - check, [418](#)
  - CheckerImplem\_fgemm, [417](#)
- CheckerImplem\_ftrsm
  - CheckerImplem\_ftrsm< Field >, [418](#)
- CheckerImplem\_ftrsm< Field >, [418](#)
  - ~CheckerImplem\_ftrsm, [418](#)
  - check, [419](#)
  - CheckerImplem\_ftrsm, [418](#)
- CheckerImplem\_invert
  - CheckerImplem\_invert< Field >, [419](#)
- CheckerImplem\_invert< Field >, [419](#)
  - ~CheckerImplem\_invert, [419](#)
  - check, [420](#)
  - CheckerImplem\_invert, [419](#)
- CheckerImplem\_PLUQ
  - CheckerImplem\_PLUQ< Field >, [420](#)
- CheckerImplem\_PLUQ< Field >, [420](#)
  - ~CheckerImplem\_PLUQ, [420](#)
  - check, [420](#)
  - CheckerImplem\_PLUQ, [420](#)
- checkers.doxy, [808](#)
- checkers\_fflas.h, [808](#)
- checkers\_fflas.inl, [808](#)
  - FFLASFFPACK\_checkers\_fflas\_inl\_H, [809](#)
- checkers\_ffpack.h, [809](#)
- checkers\_ffpack.inl, [809](#)
  - FFLASFFPACK\_checkers\_ffpack\_inl\_H, [810](#)
- checkingMessage
  - test-nullspace.C, [1094](#)
- checkMonotonicApplyP
  - test-permutations.C, [1096](#)
- checkOut

- MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, 501
- checkRPM
  - test-rpm.C, 1099
- checkSymmetricRPM
  - test-rpm.C, 1099
- checkZeroDimCharpoly
  - regression-check.C, 1054
- checkZeroDimMinPoly
  - regression-check.C, 1054
- chooseField
  - FFPACK, 390
- chooseField< Givaro::ZRing< double > >
  - FFPACK, 391
- chooseField< Givaro::ZRing< float > >
  - FFPACK, 390
- chooseField< Givaro::ZRing< int32\_t > >
  - FFPACK, 390
- chooseField< Givaro::ZRing< int64\_t > >
  - FFPACK, 390
- chunk
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, 746
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, 748
  - Sparse< \_Field, SparseMatrix\_t::SELL >, 752
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, 755
- chunkSize
  - Sparse< \_Field, SparseMatrix\_t::SELL >, 753
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, 756
- clapack.C, 1051
  - \_\_FFLASFFPACK\_CONFIGURATION, 1051
  - \_\_FFLASFFPACK\_HAVE\_CLAPACK, 1051
  - \_\_FFLASFFPACK\_HAVE\_LAPACK, 1051
  - main, 1051
- Classic, 421
- CLASSIC\_HYBRID
  - benchmark-fgemm.C, 789
- clz
  - bit\_manipulation.h, 1043
- Cmax
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, 502
- Cmin
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, 502
- cmp
  - test-simd.C, 1102
- cmp\_false
  - ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >, 560
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, 563
- cmp\_true
  - ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >, 560
- ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, 563
- col
  - Coo< Field >, 433
  - Coo< ValT, IdxT >, 431, 434
  - Sparse< \_Field, SparseMatrix\_t::COO >, 736
  - Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, 738
  - Sparse< \_Field, SparseMatrix\_t::CSR >, 740
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, 741
  - Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, 743
  - Sparse< \_Field, SparseMatrix\_t::ELL >, 745
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, 747
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, 748
  - Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, 750
  - Sparse< \_Field, SparseMatrix\_t::SELL >, 753
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, 756
- colblockindex
  - ForStrategy2D< blocksize\_t, Cut, Param >, 465
- colBlockSize
  - ForStrategy2D< blocksize\_t, Cut, Param >, 466
- coldim
  - StatsMatrix, 757
- colnumblocks
  - ForStrategy2D< blocksize\_t, Cut, Param >, 465
- ColRankProfileSubmatrix
  - FFPACK, 335, 374
- ColRankProfileSubmatrix\_modular\_double
  - ffpack.C, 1002
  - ffpack\_c.h, 1022
- ColRankProfileSubmatrixIndices
  - FFPACK, 334, 374
- ColRankProfileSubmatrixIndices\_modular\_double
  - ffpack.C, 1002
  - ffpack\_c.h, 1021
- Column, 421
- ColumnEchelonForm
  - FFPACK, 316, 317, 368
- ColumnEchelonForm\_modular\_double
  - ffpack.C, 992
  - ffpack\_c.h, 1014
- ColumnEchelonForm\_modular\_float
  - ffpack.C, 993
  - ffpack\_c.h, 1015
- ColumnEchelonForm\_modular\_int32\_t
  - ffpack.C, 994
  - ffpack\_c.h, 1015
- ColumnRankProfile
  - FFPACK, 331, 332, 373
- ColumnRankProfile\_modular\_double
  - ffpack.C, 1001
  - ffpack\_c.h, 1020
- COMMA
  - parallel.h, 1036
- CompactElement< double >, 421
  - type, 422
- CompactElement< Element >, 421



- type, [421](#)
- CompactElement< float >, [422](#)
  - type, [422](#)
- CompactElement< int16\_t >, [422](#)
  - type, [422](#)
- CompactElement< int32\_t >, [422](#)
  - type, [422](#)
- CompactElement< int64\_t >, [422](#)
  - type, [423](#)
- compatible\_data\_type< Field >, [423](#)
  - value, [423](#)
- compatible\_data\_type< Givaro::ZRing< double > >, [423](#)
  - value, [423](#)
- compatible\_data\_type< Givaro::ZRing< float > >, [423](#)
  - value, [423](#)
- compliant
  - NoSimd< T >, [521](#)
  - Simd128\_impl< true, true, false, 2 >, [569](#)
  - Simd128\_impl< true, true, false, 4 >, [579](#)
  - Simd128\_impl< true, true, false, 8 >, [589](#)
  - Simd128\_impl< true, true, true, 2 >, [597](#)
  - Simd128\_impl< true, true, true, 4 >, [606](#)
  - Simd128\_impl< true, true, true, 8 >, [615](#)
  - Simd256\_impl< true, false, true, 8 >, [626](#)
  - Simd256\_impl< true, true, false, 2 >, [636](#)
  - Simd256\_impl< true, true, false, 4 >, [651](#)
  - Simd256\_impl< true, true, false, 8 >, [665](#)
  - Simd256\_impl< true, true, true, 2 >, [672](#)
  - Simd256\_impl< true, true, true, 4 >, [682](#), [688](#)
  - Simd256\_impl< true, true, true, 8 >, [698](#)
  - Simd512\_impl< true, false, true, 8 >, [707](#)
  - Simd512\_impl< true, true, false, 8 >, [718](#)
  - Simd512\_impl< true, true, true, 8 >, [726](#)
- Compose
  - Compose< H1, H2 >, [424](#)
- Compose< H1, H2 >, [424](#)
  - Compose, [424](#)
  - first\_component, [424](#)
  - operator<<, [425](#)
  - second\_component, [424](#)
- composePermutationsLLL
  - FFPACK, [360](#)
  - ffpack.C, [989](#)
  - ffpack\_c.h, [1011](#)
- composePermutationsLLM
  - FFPACK, [360](#)
  - ffpack.C, [989](#)
  - ffpack\_c.h, [1010](#)
- composePermutationsMLM
  - FFPACK, [361](#)
  - ffpack.C, [989](#)
  - ffpack\_c.h, [1011](#)
- CompressRows
  - FFPACK::Protected, [396](#)
- CompressRowsQA
  - FFPACK::Protected, [397](#)
- CompressRowsQK
  - FFPACK::Protected, [397](#)
- CompressToBlockBiDiagonal
  - FFPACK, [344](#)
- computeDeviation
  - FFLAS, [153](#)
- computeFactorClassic
  - FFLAS::Protected, [212](#)
- ComputeRPermutation
  - FFPACK, [346](#), [348](#)
- computeS1S2
  - FFLAS, [123](#)
- config-blas.h, [810](#)
  - AtlasConj, [812](#)
  - blas\_enum, [811](#)
  - CBLAS\_DIAG, [812](#)
  - cblas\_dsyrk, [816](#)
  - CBLAS\_ENUM\_DEFINED\_H, [811](#)
  - CBLAS\_EXTERNALS, [811](#)
  - CBLAS\_INT, [811](#)
  - CBLAS\_ORDER, [811](#)
  - CBLAS\_SIDE, [812](#)
  - CBLAS\_TRANSPOSE, [811](#)
  - CBLAS\_UPLO, [812](#)
  - CblasColMajor, [811](#)
  - CblasConjTrans, [812](#)
  - CblasLeft, [812](#)
  - CblasLower, [812](#)
  - CblasNonUnit, [812](#)
  - CblasNoTrans, [812](#)
  - CblasRight, [812](#)
  - CblasRowMajor, [811](#)
  - CblasTrans, [812](#)
  - CblasUnit, [812](#)
  - CblasUpper, [812](#)
  - dasum\_, [813](#)
  - daxpy\_, [812](#)
  - dcopy\_, [814](#)
  - ddot\_, [812](#)
  - dgemm\_, [816](#)
  - dgemv\_, [813](#)
  - dger\_, [814](#)
  - dnrm2\_, [813](#)
  - dscal\_, [814](#)
  - dtrmm\_, [815](#)
  - dtrsm\_, [815](#)
  - idamax\_, [813](#)
  - saxpy\_, [812](#)
  - scopy\_, [814](#)
  - sdot\_, [813](#)
  - sgemm\_, [816](#)
  - sgemv\_, [813](#)
  - sger\_, [814](#)
  - sscal\_, [815](#)
  - strmm\_, [815](#)
  - strsm\_, [815](#)
- config.h, [817](#), [820](#)
  - \_\_FFLASFFPACK\_HAVE\_BLAS, [821](#)
  - \_\_FFLASFFPACK\_HAVE\_CBLAS, [821](#)



- \_\_FFLASFFPACK\_HAVE\_CXX11, [821](#)
- \_\_FFLASFFPACK\_HAVE\_DLFCN\_H, [821](#)
- \_\_FFLASFFPACK\_HAVE\_FLOAT\_H, [821](#)
- \_\_FFLASFFPACK\_HAVE\_INT128, [821](#)
- \_\_FFLASFFPACK\_HAVE\_INTPTR\_T, [821](#)
- \_\_FFLASFFPACK\_HAVE\_LAPACK, [821](#)
- \_\_FFLASFFPACK\_HAVE\_LIMITS\_H, [821](#)
- \_\_FFLASFFPACK\_HAVE\_LITTLE\_ENDIAN, [821](#)
- \_\_FFLASFFPACK\_HAVE\_PTHREAD\_H, [821](#)
- \_\_FFLASFFPACK\_HAVE\_STDDEF\_H, [821](#)
- \_\_FFLASFFPACK\_HAVE\_STDINT\_H, [822](#)
- \_\_FFLASFFPACK\_HAVE\_STDIO\_H, [822](#)
- \_\_FFLASFFPACK\_HAVE\_STDLIB\_H, [822](#)
- \_\_FFLASFFPACK\_HAVE\_STRINGS\_H, [822](#)
- \_\_FFLASFFPACK\_HAVE\_STRING\_H, [822](#)
- \_\_FFLASFFPACK\_HAVE\_SYS\_STAT\_H, [822](#)
- \_\_FFLASFFPACK\_HAVE\_SYS\_TIME\_H, [822](#)
- \_\_FFLASFFPACK\_HAVE\_SYS\_TYPES\_H, [822](#)
- \_\_FFLASFFPACK\_HAVE\_UNISTD\_H, [822](#)
- \_\_FFLASFFPACK\_LT\_OBJDIR, [822](#)
- \_\_FFLASFFPACK\_OPENBLAS\_NUM\_THREADS, [822](#)
- \_\_FFLASFFPACK\_PACKAGE, [822](#)
- \_\_FFLASFFPACK\_PACKAGE\_BUGREPORT, [822](#)
- \_\_FFLASFFPACK\_PACKAGE\_NAME, [822](#)
- \_\_FFLASFFPACK\_PACKAGE\_STRING, [822](#)
- \_\_FFLASFFPACK\_PACKAGE\_TARNAME, [823](#)
- \_\_FFLASFFPACK\_PACKAGE\_URL, [823](#)
- \_\_FFLASFFPACK\_PACKAGE\_VERSION, [823](#)
- \_\_FFLASFFPACK\_SIZEOF\_CHAR, [823](#)
- \_\_FFLASFFPACK\_SIZEOF\_INT, [823](#)
- \_\_FFLASFFPACK\_SIZEOF\_LONG, [823](#)
- \_\_FFLASFFPACK\_SIZEOF\_LONG\_LONG, [823](#)
- \_\_FFLASFFPACK\_SIZEOF\_SHORT, [823](#)
- \_\_FFLASFFPACK\_SIZEOF\_\_INT64\_T, [823](#)
- \_\_FFLASFFPACK\_STDC\_HEADERS, [823](#)
- \_\_FFLASFFPACK\_USE\_OPENMP, [823](#)
- \_\_FFLASFFPACK\_VERSION, [823](#)
- HAVE\_BLAS, [817](#)
- HAVE\_CBLAS, [817](#)
- HAVE\_CXX11, [817](#)
- HAVE\_DLFCN\_H, [818](#)
- HAVE\_FLOAT\_H, [818](#)
- HAVE\_INT128, [818](#)
- HAVE\_INTPTR\_T, [818](#)
- HAVE\_LAPACK, [818](#)
- HAVE\_LIMITS\_H, [818](#)
- HAVE\_LITTLE\_ENDIAN, [818](#)
- HAVE\_PTHREAD\_H, [818](#)
- HAVE\_STDDEF\_H, [818](#)
- HAVE\_STDINT\_H, [818](#)
- HAVE\_STDIO\_H, [818](#)
- HAVE\_STDLIB\_H, [818](#)
- HAVE\_STRING\_H, [818](#)
- HAVE\_STRINGS\_H, [818](#)
- HAVE\_SYS\_STAT\_H, [818](#)
- HAVE\_SYS\_TIME\_H, [819](#)
- HAVE\_SYS\_TYPES\_H, [819](#)
- HAVE\_UNISTD\_H, [819](#)
- LT\_OBJDIR, [819](#)
- OPENBLAS\_NUM\_THREADS, [819](#)
- PACKAGE, [819](#)
- PACKAGE\_BUGREPORT, [819](#)
- PACKAGE\_NAME, [819](#)
- PACKAGE\_STRING, [819](#)
- PACKAGE\_TARNAME, [819](#)
- PACKAGE\_URL, [819](#)
- PACKAGE\_VERSION, [819](#)
- SIZEOF\_\_INT64\_T, [820](#)
- SIZEOF\_CHAR, [819](#)
- SIZEOF\_INT, [819](#)
- SIZEOF\_LONG, [819](#)
- SIZEOF\_LONG\_LONG, [820](#)
- SIZEOF\_SHORT, [820](#)
- STDC\_HEADERS, [820](#)
- USE\_OPENMP, [820](#)
- VERSION, [820](#)
- Configuring and Installing FFLAS-FFPACK, [2](#)
- CONST
  - fflas\_simd.h, [872](#)
- ConstElement\_ptr
  - benchmark-fgemv-rns.C, [788](#)
  - rns\_double, [528](#)
  - rns\_double\_extended, [539](#)
  - RNSInteger< RNS >, [543](#)
  - RNSIntegerMod< RNS >, [547](#)
- CONSTREFERENCE
  - parallel.h, [1034](#)
- convert
  - rns\_double, [529](#), [530](#)
  - rns\_double\_extended, [540](#)
  - RNSInteger< RNS >, [544](#)
  - RNSIntegerMod< RNS >, [549](#)
- convert\_transpose
  - rns\_double, [529](#)
- ConvertTo< T >, [430](#)
- COO
  - FFLAS, [76](#)
- Coo
  - Coo< Field >, [432](#)
  - Coo< ValT, IdxT >, [431](#), [434](#)
- coo
  - HelperFlag, [474](#)
- Coo< Field >, [432](#)
  - col, [433](#)
  - Coo, [432](#)
  - deleted, [433](#)
  - operator=, [433](#)
  - row, [433](#)
  - val, [433](#)
- Coo< ValT, IdxT >, [430](#), [433](#)
  - col, [431](#), [434](#)
  - Coo, [431](#), [434](#)
  - operator=, [431](#), [434](#)
  - row, [431](#), [434](#)
  - Self, [431](#), [434](#)

- val, [431](#), [434](#)
- coo.h, [888](#)
- coo\_spm.inl, [888](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_coo\_spm.inl, [889](#)
- coo\_spmv.inl, [889](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_coo\_spmv.inl, [890](#)
- coo\_utils.inl, [890](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_coo\_utils.inl, [890](#)
- COO\_ZO
  - FFLAS, [76](#)
- CooMat< Field >, [435](#)
  - \_coo16, [435](#)
  - \_coo16\_zo, [435](#)
  - \_coo32, [435](#)
  - \_coo32\_zo, [435](#)
  - \_coo64, [435](#)
  - \_coo64\_zo, [435](#)
- Copying and Licence, [2](#)
- count\_nonconst\_lvalue\_reference< const T &, O... >, [436](#)
  - n, [436](#)
- count\_nonconst\_lvalue\_reference< T >, [436](#)
- count\_nonconst\_lvalue\_reference< T &, O... >, [436](#)
  - n, [436](#)
- count\_nonconst\_lvalue\_reference< T, O... >, [436](#)
  - n, [436](#)
- count\_nonconst\_lvalue\_reference<>, [437](#)
  - n, [437](#)
- CROUT
  - ffpack\_pluq.inl, [942](#)
- CSC
  - FFLAS, [76](#)
- CSC\_ZO
  - FFLAS, [76](#)
- CSR
  - FFLAS, [76](#)
- csr
  - HelperFlag, [474](#)
- csr.h, [890](#)
- CSR\_HYB
  - FFLAS, [76](#)
- csr\_hyb.h, [895](#)
- csr\_hyb\_pspmm.inl, [895](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_pspmm.inl, [896](#)
- csr\_hyb\_pspmv.inl, [896](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_pspmv.inl, [896](#)
- csr\_hyb\_spm.inl, [897](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_spm.inl, [897](#)
- csr\_hyb\_spmv.inl, [897](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_spmv.inl, [897](#)
- csr\_hyb\_utils.inl, [898](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_utils.inl, [898](#)
- csr\_pspmm.inl, [891](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_pspmm.inl, [892](#)
- csr\_pspmv.inl, [892](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_pspmv.inl, [892](#)
- csr\_spm.inl, [892](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_spm.inl, [893](#)
- csr\_spmv.inl, [894](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_spmv.inl, [894](#)
- csr\_utils.inl, [894](#)
- CSR\_ZO
  - FFLAS, [76](#)
- CsrMat< Field >, [437](#)
  - \_csr16, [437](#)
  - \_csr16\_zo, [437](#)
  - \_csr32, [437](#)
  - \_csr32\_zo, [437](#)
  - \_csr64, [437](#)
  - \_csr64\_zo, [437](#)
- cst
  - Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, [738](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, [743](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, [747](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, [749](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, [755](#)
- ctz
  - bit\_manipulation.h, [1043](#)
- CUBE
  - benchmark-checkers.C, [781](#)
  - benchmark-fsytrf.C, [796](#)
  - benchmark-ftsrf.C, [798](#)
  - benchmark-inverse.C, [799](#)
  - benchmark-lqup.C, [800](#)
  - benchmark-pluq.C, [801](#)
  - benchmark-wino.C, [803](#)
  - charpoly.C, [774](#)
  - fsyrk.C, [775](#)
  - fsytrf.C, [776](#)
  - ftsrf.C, [776](#)
  - pluq.C, [777](#)
  - mda.C, [1052](#)
  - main, [1052](#)
- current
  - ForStrategy1D< blocksize\_t, Cut, Param >, [462](#)
  - ForStrategy2D< blocksize\_t, Cut, Param >, [466](#)
- Cut
  - Parallel< C, P >, [522](#)
- cyclic\_shift\_col
  - FFPACK, [362](#), [365](#)
- cyclic\_shift\_col\_modular\_double
  - ffpack.C, [990](#)
  - ffpack\_c.h, [1011](#)

- cyclic\_shift\_mathPerm
  - FFPACK, [361](#)
  - ffpack.C, [989](#)
  - ffpack\_c.h, [1011](#)
- cyclic\_shift\_row
  - FFPACK, [361](#), [364](#)
- cyclic\_shift\_row\_col
  - FFPACK, [361](#), [364](#)
- cyclic\_shift\_row\_modular\_double
  - ffpack.C, [989](#)
  - ffpack\_c.h, [1011](#)
- Danilevski
  - FFPACK, [350](#)
  - FFPACK::Protected, [395](#)
- dasum\_
  - config-blas.h, [813](#)
- dat
  - Sparse< \_Field, SparseMatrix\_t::COO >, [736](#)
  - Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, [738](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR >, [740](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, [741](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, [744](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL >, [745](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, [747](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, [748](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, [750](#)
  - Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, [751](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL >, [754](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, [756](#)
- data
  - Argument, [404](#)
- daxpy\_
  - config-blas.h, [812](#)
- dcopy\_
  - config-blas.h, [814](#)
- ddot\_
  - config-blas.h, [812](#)
- debug.h, [1043](#)
  - FFLASFFPACK\_abort, [1044](#)
  - FFLASFFPACK\_check, [1044](#)
- DeCompressRows
  - FFPACK::Protected, [397](#)
- DeCompressRowsQA
  - FFPACK::Protected, [398](#)
- DeCompressRowsQK
  - FFPACK::Protected, [397](#)
- DefaultBoundedTag, [438](#)
- DefaultTag, [438](#)
- delayed
  - RNSIntegerMod< RNS >, [547](#)
  - Sparse< \_Field, SparseMatrix\_t::COO >, [736](#)
  - Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, [738](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR >, [739](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, [741](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, [743](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL >, [744](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, [746](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, [747](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, [749](#)
  - Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, [751](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL >, [752](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, [755](#)
- DelayedField
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [499](#)
- delayedField
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [503](#)
- DelayedField\_t
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [499](#)
- DelayedTag, [438](#)
- deleted
  - Coo< Field >, [433](#)
- DENSE\_THRESHOLD
  - fflas\_sparse.h, [885](#)
- denseCols
  - StatsMatrix, [759](#)
- denseRows
  - StatsMatrix, [759](#)
- Det
  - FFPACK, [327](#), [328](#), [351](#), [352](#), [371](#), [372](#)
- det.C, [804](#)
  - main, [804](#)
- Det\_modular\_double
  - ffpack.C, [999](#)
  - ffpack\_c.h, [1019](#)
- deviationCol
  - StatsMatrix, [758](#)
- deviationColDifference
  - StatsMatrix, [758](#)
- deviationRow
  - StatsMatrix, [758](#)
- deviationRowDifference
  - StatsMatrix, [758](#)
- DFelt
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [499](#)
- dgemm\_
  - config-blas.h, [816](#)
  - fblas.C, [1052](#)
- dgemv\_
  - config-blas.h, [813](#)
- dger\_
  - config-blas.h, [814](#)
- digits
  - limits< char >, [488](#)
  - limits< double >, [489](#)
  - limits< float >, [489](#)
  - limits< int >, [491](#)
  - limits< long >, [491](#)
  - limits< long long >, [492](#)

- limits< short int >, 494
- limits< signed char >, 494
- limits< unsigned char >, 495
- limits< unsigned int >, 496
- limits< unsigned long >, 496
- limits< unsigned long long >, 497
- limits< unsigned short int >, 497
- div
  - ScalFunctions< Element >, 556
  - Simd256\_impl< true, false, true, 8 >, 628
  - Simd512\_impl< true, false, true, 8 >, 710
- DivideAndConquer, 438
- dnrm2\_
  - config-blas.h, 813
- DNS\_BIN\_VER
  - read\_sparse.h, 909
- doApplyS
  - FFPACK, 357
- doApplyT
  - FFPACK, 358
- doBenchs
  - Bench< Elt >, 408
- doTests
  - Test< Elt >, 763
- DotProdBoundClassic
  - FFLAS::Protected, 212
- DOUBLE\_TO\_FLOAT\_CROSSOVER
  - fflas.h, 826
  - winograd.C, 778
- dscal\_
  - config-blas.h, 814
- dtrmm\_
  - config-blas.h, 815
- dtrsm\_
  - config-blas.h, 815
- DynamicPeeling
  - FFLAS::Protected, 217
- DynamicPeeling2
  - FFLAS::Protected, 217
- EFFGFF
  - benchmark-dsytrf.C, 783
- Element
  - FieldSimd< \_Field >, 447
  - readMyMachineType< Field, mpz\_t >, 526
  - readMyMachineType< Field, T >, 525
  - rns\_double, 528
  - rns\_double\_extended, 539
  - RNSInteger< RNS >, 543
  - RNSIntegerMod< RNS >, 547
  - TestOneMethod< Simd >, 765
- Element\_ptr
  - benchmark-fgemm-rns.C, 788
  - readMyMachineType< Field, mpz\_t >, 526
  - readMyMachineType< Field, T >, 525
  - rns\_double, 528
  - rns\_double\_extended, 539
  - RNSInteger< RNS >, 543
  - RNSIntegerMod< RNS >, 547
- ElementTraits< double >, 439
  - value, 439
- ElementTraits< Element >, 438
  - value, 439
- ElementTraits< FFPACK::rns\_double\_elt >, 439
  - value, 439
- ElementTraits< float >, 439
  - value, 439
- ElementTraits< Givaro::Integer >, 440
  - value, 440
- ElementTraits< int16\_t >, 440
  - value, 440
- ElementTraits< int32\_t >, 440
  - value, 440
- ElementTraits< int64\_t >, 440
  - value, 441
- ElementTraits< int8\_t >, 441
  - value, 441
- ElementTraits< RecInt::rint< K > >, 441
  - value, 441
- ElementTraits< RecInt::rmint< K, MG > >, 441
  - value, 441
- ElementTraits< RecInt::ruint< K > >, 442
  - value, 442
- ElementTraits< uint16\_t >, 442
  - value, 442
- ElementTraits< uint32\_t >, 442
  - value, 442
- ElementTraits< uint64\_t >, 442
  - value, 443
- ElementTraits< uint8\_t >, 443
  - value, 443
- ELL
  - FFLAS, 76
- ell
  - HelperFlag, 474
- ell.h, 898
- ell\_pspmm.inl, 899
  - \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_pspmm\_INL, 899
- ell\_pspmv.inl, 899
  - \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_pspmv\_INL, 900
- ELL\_simd
  - FFLAS, 76
- ell\_simd.h, 902
- ell\_simd\_pspmv.inl, 903
  - \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_simd\_pspmv\_INL, 904
- ell\_simd\_spmv.inl, 904
  - \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_simd\_spmv\_INL, 904
- ell\_simd\_utils.inl, 905
  - \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_simd\_utils\_INL, 905
- ELL\_simd\_ZO
  - FFLAS, 76
- ell\_spmmm.inl, 900

- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_spmv\_INL, 901
- ell\_spmv.inl, 901
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_spmv\_INL, 902
- ell\_utils.inl, 902
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_utils\_INL, 902
- ELL\_ZO
  - FFLAS, 76
- EllMat< Field >, 443
  - \_ell16, 443
  - \_ell16\_zo, 443
  - \_ell32, 443
  - \_ell32\_zo, 444
  - \_ell64, 443
  - \_ell64\_zo, 444
- Elt\_ptr
  - Bench< Elt >, 407
  - Test< Elt >, 762
- ENABLE\_ALL\_CHECKINGS
  - benchmark-checkers.C, 781
  - ffpack\_ftrtr.inl, 936
  - test-fdot.C, 1062
  - test-fgemm-check.C, 1063
  - test-fsyr2k.C, 1074
  - test-fsyrk.C, 1075
  - test-ftmv.C, 1079
  - test-ftsm-check.C, 1080
  - test-ftsm.C, 1081
  - test-ftssyr2k.C, 1082
  - test-ftrstr.C, 1083
  - test-ftrsv.C, 1085
  - test-ftrtri.C, 1086
  - test-invert-check.C, 1087
  - test-pluq-check.C, 1097
- ENABLE\_CHECKER\_charpoly
  - test-charpoly-check.C, 1054
- ENABLE\_CHECKER\_Det
  - test-det-check.C, 1057
- ENABLE\_CHECKER\_fgemm
  - test-fgemm.C, 1065
- enable\_if\_no\_simd\_t
  - Bench< Elt >, 407
  - Test< Elt >, 762
- enable\_if\_simd128\_t
  - Bench< Elt >, 407
  - Test< Elt >, 762
- enable\_if\_simd256\_t
  - Bench< Elt >, 407
  - Test< Elt >, 762
- enable\_if\_simd512\_t
  - Bench< Elt >, 407
  - Test< Elt >, 763
- enable\_if\_t
  - Bench< Elt >, 407
  - Test< Elt >, 762
  - TestOneMethod< Simd >, 765
- end
  - ForStrategy1D< blocksize\_t, Cut, Param >, 462
- END\_OF\_ARGUMENTS
  - args-parser.h, 1041
- END\_PARALLEL\_MAIN
  - parallel.h, 1034
- eq
  - ScalFunctions< Element >, 557
  - Simd128\_impl< true, true, false, 2 >, 573
  - Simd128\_impl< true, true, false, 4 >, 583
  - Simd128\_impl< true, true, false, 8 >, 593
  - Simd128\_impl< true, true, true, 2 >, 602
  - Simd128\_impl< true, true, true, 4 >, 611
  - Simd128\_impl< true, true, true, 8 >, 620
  - Simd256\_impl< true, false, true, 8 >, 629
  - Simd256\_impl< true, true, false, 2 >, 640
  - Simd256\_impl< true, true, false, 4 >, 658
  - Simd256\_impl< true, true, false, 8 >, 668
  - Simd256\_impl< true, true, true, 2 >, 677
  - Simd256\_impl< true, true, true, 4 >, 687, 694
  - Simd256\_impl< true, true, true, 8 >, 703
  - Simd512\_impl< true, false, true, 8 >, 711
  - Simd512\_impl< true, true, false, 8 >, 722
  - Simd512\_impl< true, true, true, 8 >, 731
- eval\_func\_on\_array
  - test-simd.C, 1102
- evaluate\_scalar\_method
  - TestOneMethod< Simd >, 765, 766
- evaluate\_simd\_method
  - TestOneMethod< Simd >, 766
- example
  - Argument, 403
- ExpandBlockBiDiagonalToBruhat
  - FFPACK, 344
- expandLCRE
  - FFPACK, 349
- F
  - Bench< Elt >, 408
  - Test< Elt >, 764
- fadd
  - FFLAS, 77, 78, 80, 82, 163, 164, 171, 172
  - FFLAS::details, 199, 200
- fadd\_1\_modular\_double
  - fflas\_c.h, 957
  - fflas\_lvl1.C, 976
- fadd\_2\_modular\_double
  - fflas\_c.h, 960
  - fflas\_lvl2.C, 981
- faddin
  - FFLAS, 78, 81, 164, 172
- faddin\_1\_modular\_double
  - fflas\_c.h, 957
  - fflas\_lvl1.C, 977
- faddin\_2\_modular\_double
  - fflas\_c.h, 961
  - fflas\_lvl2.C, 981
- Failure, 444
  - \_errorStream, 445

- Failure, 444
- operator(), 444, 445
- print, 445
- setErrorStream, 445
- failure
  - FFPACK, 377
- FailureCharpolyCheck, 445
- FailureDetCheck, 446
- FailureFgemmCheck, 446
- FailureInvertCheck, 446
- FailurePLUQCheck, 446
- FailureTrsmCheck, 446
- fassign
  - FFLAS, 82–84, 160, 164
- fassign\_1\_modular\_double
  - fflas\_c.h, 955
  - fflas\_lvl1.C, 975
- fassign\_2\_modular\_double
  - fflas\_c.h, 957
  - fflas\_lvl2.C, 978
- faxpby
  - FFLAS, 132, 139
- faxpy
  - FFLAS, 84, 85, 162, 169
  - FFLAS::details, 201
- faxpy\_1\_modular\_double
  - fflas\_c.h, 956
  - fflas\_lvl1.C, 976
- faxpy\_2\_modular\_double
  - fflas\_c.h, 960
  - fflas\_lvl2.C, 980
- fblas.C, 1052
  - \_\_FFLASFFPACK\_CONFIGURATION, 1052
  - dgemm\_, 1052
  - main, 1052
- fconvert
  - FFLAS, 129, 137, 158
- fconvert\_rns
  - FFLAS, 155, 156
- fconvert\_trans\_rns
  - FFLAS, 156
- fdot
  - FFLAS, 86–88, 133, 162, 181
- fdot\_1\_modular\_double
  - fflas\_c.h, 956
  - fflas\_lvl1.C, 976
- fequal
  - FFLAS, 132, 136, 160, 165
- fequal\_1\_modular\_double
  - fflas\_c.h, 955
  - fflas\_lvl1.C, 975
- fequal\_2\_modular\_double
  - fflas\_c.h, 958
  - fflas\_lvl2.C, 978
- FFLAS, 41, 45
  - alignable, 188
  - alignable< Givaro::Integer \* >, 188
  - BaseTimer, 74
  - bitsize, 140
  - bitsize< Givaro::ZRing< Givaro::Integer > >, 140
  - BlockCuts, 179, 181
  - BlockCuts< CuttingStrategy::Block, StrategyParameter::Fixed >, 180
  - BlockCuts< CuttingStrategy::Block, StrategyParameter::Grain >, 179
  - BlockCuts< CuttingStrategy::Block, StrategyParameter::Threads >, 181
  - BlockCuts< CuttingStrategy::Column, StrategyParameter::Fixed >, 180
  - BlockCuts< CuttingStrategy::Column, StrategyParameter::Grain >, 180
  - BlockCuts< CuttingStrategy::Column, StrategyParameter::Threads >, 180
  - BlockCuts< CuttingStrategy::Row, StrategyParameter::Fixed >, 179
  - BlockCuts< CuttingStrategy::Row, StrategyParameter::Grain >, 179
  - BlockCuts< CuttingStrategy::Row, StrategyParameter::Threads >, 180
  - BlockCuts< CuttingStrategy::Single, StrategyParameter::Threads >, 179
  - cblas\_imptrsm, 127
  - Checker\_fgemm, 72
  - Checker\_ftrsm, 72
  - computeDeviation, 153
  - computeS1S2, 123
  - COO, 76
  - COO\_ZO, 76
  - CSC, 76
  - CSC\_ZO, 76
  - CSR, 76
  - CSR\_HYB, 76
  - CSR\_ZO, 76
  - ELL, 76
  - ELL\_simd, 76
  - ELL\_simd\_ZO, 76
  - ELL\_ZO, 76
  - fadd, 77, 78, 80, 82, 163, 164, 171, 172
  - faddin, 78, 81, 164, 172
  - fassign, 82–84, 160, 164
  - faxpby, 132, 139
  - faxpy, 84, 85, 162, 169
  - fconvert, 129, 137, 158
  - fconvert\_rns, 155, 156
  - fconvert\_trans\_rns, 156
  - fdot, 86–88, 133, 162, 181
  - fequal, 132, 136, 160, 165
  - FFLAS\_BASE, 75
  - fflas\_delete, 154, 188
  - FFLAS\_DIAG, 75
  - FFLAS\_FORMAT, 76
  - fflas\_new, 155, 156, 188
  - FFLAS\_ORDER, 74
  - FFLAS\_SIDE, 75
  - FFLAS\_TRANSPOSE, 74
  - FFLAS\_UPLO, 75

- FflasAuto, 76
- FflasBinary, 76
- FflasColMajor, 74
- FflasDense, 76
- FflasDouble, 76
- FflasFloat, 76
- FflasGeneric, 76
- FflasLeft, 75
- FflasLeftTri, 75
- FflasLower, 75
- FflasMaple, 76
- FflasMath, 76
- FflasNonUnit, 75
- FflasNoTrans, 75
- FflasRight, 75
- FflasRightTri, 75
- FflasRowMajor, 74
- FflasSageMath, 76
- FflasSMS, 76
- FflasTrans, 75
- FflasUnit, 75
- FflasUpper, 75
- fgemm, 88–90, 92–97, 144, 176–178
- fgemv, 97–102, 173, 184
- fger, 103–106, 173
- fidentity, 136, 137, 166
- finit, 108, 110, 129, 137, 157, 167
- finit\_rns, 155, 156
- finit\_trans\_rns, 155
- fiszero, 131, 136, 159, 166
- fmove, 139, 170
- fneg, 130, 138, 159, 168
- fnegin, 130, 138, 158, 168
- ForceCheck\_fgemm, 72
- ForceCheck\_ftrsm, 72
- frand, 131, 135
- freduce, 107–111, 157, 167
- freduce\_constoverride, 108, 110
- freivalds, 111
- fscal, 112–116, 161, 169
- fscaln, 111, 113–116, 161, 169
- fspmm, 145
- fspmv, 145, 152
- fsquare, 91, 92, 178
- fsub, 78, 80, 164, 171
- fsubin, 78, 81, 172
- fswap, 133, 163
- fsyr2k, 116
- fsyrk, 117–124
- fsyrk\_strassen, 124, 142
- ftrmm, 124, 125, 175
- ftrmv, 140
- ftrsm, 126, 127, 141, 144, 145, 175
- ftrsv, 128, 174
- fzero, 130, 133, 135, 159, 165
- getArgumentValue, 185
- getDataType, 151, 152
- getSeed, 189
- getStat, 154
- getTLBSize, 189
- has\_equal, 73
- has\_minus, 73
- has\_minus\_eq, 73
- has\_mul, 73
- has\_mul\_eq, 74
- has\_plus, 73
- has\_plus\_eq, 73
- HYB\_ZO, 76
- igemm\_, 128
- InfNorm, 77
- max3, 77
- max4, 77
- maxCardinality, 154
- maxCardinality< Givaro::Modular< int32\_t > >, 154
- maxCardinality< Givaro::Modular< int64\_t > >, 154
- min3, 77
- min4, 77
- minCardinality, 154
- MKLSparseMatrixFormat, 73
- mone, 76
- NoSimdSparseMatrix, 73
- NotMKLSparseMatrixFormat, 73
- NotZOSparseMatrix, 72
- number\_kind, 76
- one, 76
- operator<<, 151
- other, 76
- parseArguments, 185
- pfadd, 79
- pfaddin, 79
- pfgemm, 142, 182–184
- pfgemm\_1D\_rec, 142
- pfgemm\_2D\_rec, 143
- pfgemm\_3D\_rec, 143
- pfgemm\_3D\_rec2, 144
- pfrand, 181
- pfreduce, 109
- pfsb, 79
- pfsbin, 80
- pfzero, 181
- preamble, 186
- prefetch, 189
- queryCacheSizes, 189
- queryL1CacheSize, 189
- queryTopLevelCacheSize, 189
- readDnsFormat, 152
- readMachineType, 152
- ReadMatrix, 186
- readSmsFormat, 151
- readSprFormat, 151
- SELL, 76
- SELL\_ZO, 76
- SimdSparseMatrix, 72
- sparse\_delete, 146–150, 153



- sparse\_init, 145–150, 153
- sparse\_print, 147, 150, 153
- SparseMatrix\_t, 76
- SysTimer, 74
- Timer, 74
- UserTimer, 74
- writeCommandString, 185
- writeDnsFormat, 152
- WriteMatrix, 185, 187
- WritePermutation, 187
- zero, 76
- ZOSparseMatrix, 72
- fflas-101\_1.C, 1106
  - main, 1106
- fflas-101\_3.C, 1106
  - main, 1107
- FFLAS-FFPACK, 41
- FFLAS-FFPACK Documentation., 1
- FFLAS-FFPACK fields, 43
- fflas-ffpack-config.h, 823
  - GCC\_VERSION, 824
- fflas-ffpack-default-thresholds.h, 824
  - \_\_FFLASFFPACK\_ARITHPROG\_THRESHOLD, 825
  - \_\_FFLASFFPACK\_CHARPOLY\_Danilevskii\_LUKrylov\_THRESHOLD, 824
  - \_\_FFLASFFPACK\_CHARPOLY\_LUKrylov\_ArithProg\_THRESHOLD, 824
  - \_\_FFLASFFPACK\_FSYRK\_THRESHOLD, 825
  - \_\_FFLASFFPACK\_FSYTRF\_THRESHOLD, 825
  - \_\_FFLASFFPACK\_FTRTRI\_THRESHOLD, 825
  - \_\_FFLASFFPACK\_PLUQ\_THRESHOLD, 824
  - \_\_FFLASFFPACK\_WINOTHRESHOLD, 824
  - \_\_FFLASFFPACK\_WINOTHRESHOLD\_BAL, 824
  - \_\_FFLASFFPACK\_WINOTHRESHOLD\_BAL\_FLT, 824
  - \_\_FFLASFFPACK\_WINOTHRESHOLD\_FLT, 824
- fflas-ffpack-thresholds.h, 825
- fflas-ffpack.doxy, 825
- fflas-ffpack.h, 825
- fflas.doxy, 825
- fflas.h, 825
  - DOUBLE\_TO\_FLOAT\_CROSSOVER, 826
  - WINOTHRESHOLD, 826
- FFLAS::ftranspose\_impl, 190
  - nonsquare\_inplace\_v1, 190
  - nonsquare\_inplace\_v2, 190
  - not\_inplace, 190
  - square\_inplace, 190
- FFLAS::BLAS3, 191
  - Bini, 192
  - Winograd, 193
  - Winograd\_L\_S, 196
  - Winograd\_LR\_S, 196
  - Winograd\_R\_S, 197
  - WinogradAcc\_2\_24, 194
  - WinogradAcc\_2\_27, 194
  - WinogradAcc\_3\_21, 194
  - WinogradAcc\_3\_23, 193
  - WinogradAcc\_L\_S, 196
  - WinogradAcc\_LR, 195
  - WinogradAcc\_R\_S, 195
  - WinoPar, 193
- FFLAS::csr\_hyb\_details, 197
- FFLAS::CuttingStrategy, 197
  - RNSModulus, 198
- FFLAS::details, 198
  - BlockingFactor, 206
  - fadd, 199, 200
  - faxpy, 201
  - freduce, 201, 202
  - fscal, 202, 203
  - fscaln, 202, 203
  - gebp, 206
  - igebb11, 205
  - igebb14, 204
  - igebb21, 204
  - igebb24, 203
  - igebb41, 204
  - igebb44, 203
  - igebp, 205
  - pack\_lhs, 205
  - pack\_rhs, 205
  - FFLAS::details\_spmv, 206
- FFLAS::FieldCategories, 207
- FFLAS::FieldCategories, 207
- FFLAS::MMHelperAlgo, 207
- FFLAS::ModeCategories, 208
- FFLAS::ParSeqHelper, 208
- FFLAS::Protected, 209
  - computeFactorClassic, 212
  - DotProdBoundClassic, 212
  - DynamicPeeling, 217
  - DynamicPeeling2, 217
  - fgemm\_convert, 213
  - fgemv\_convert, 218
  - fger\_convert, 218
  - fsquareCommon, 215
  - fsyrk\_convert, 218
  - igemm, 222
  - igemm\_colmajor, 221, 222
  - MatF2MatD\_Triangular, 222
  - MatF2MatFI\_Triangular, 223
  - min\_types, 220, 221
  - NeedDoublePreAddReduction, 214, 215
  - NeedPreAddReduction, 213, 214
  - NeedPreAxyReduction, 219, 220
  - NeedPreScalReduction, 219
  - NeedPreSubReduction, 214
  - ScalAndReduce, 215, 219
  - TRSMBound, 212, 213
  - unfit, 221
  - WinogradCalc, 218
  - WinogradSteps, 216
  - WinogradThreshold, 216
- FFLAS::sell\_details, 223



- FFLAS::sparse\_details, 223
  - fspmm, 230–232
  - fspmm\_dispatch, 229
  - fspmv, 227–229, 236, 237
  - fspmv\_dispatch, 227
  - init\_y, 226
  - pfspmm, 233–235
  - pfspmm\_dispatch, 232
  - pfspmv, 235, 236
- FFLAS::sparse\_details\_impl, 237
  - fspmm, 245, 246, 253, 254, 259, 263, 264, 272, 273
  - fspmm\_mone, 247, 254, 264, 265
  - fspmm\_mone\_simd\_aligned, 248, 255, 266
  - fspmm\_mone\_simd\_unaligned, 248, 255, 266
  - fspmm\_one, 247, 254, 264, 265
  - fspmm\_one\_simd\_aligned, 247, 255, 265
  - fspmm\_one\_simd\_unaligned, 247, 255, 265
  - fspmm\_simd\_aligned, 246, 253
  - fspmm\_simd\_unaligned, 246, 254
  - fspmv, 248, 249, 256, 260, 266, 267, 269, 270, 273–276
  - fspmv\_mone, 249, 256, 257, 267, 270, 271, 276, 277
  - fspmv\_mone\_simd, 271, 277
  - fspmv\_one, 249, 256, 257, 267, 270, 276, 277
  - fspmv\_one\_simd, 271, 277
  - fspmv\_simd, 269, 270, 275, 276
  - pfspmm, 250, 257, 258, 260, 261, 271, 272
  - pfspmm\_mone, 251
  - pfspmm\_one, 250, 251
  - pfspmm\_zo, 262
  - pfspmv, 251, 252, 258, 259, 262, 268, 272, 274
  - pfspmv\_mone, 252, 253, 263, 268, 269, 275
  - pfspmv\_one, 252, 253, 263, 268, 269, 274, 275
  - pfspmv\_task, 252
- FFLAS::StrategyParameter, 278
- FFLAS::StructureHelper, 278
- FFLAS::vectorised, 278
  - add, 281
  - addp, 280
  - axpyp, 281
  - modp, 283
  - reduce, 281–283
  - scalp, 283, 284
  - sub, 281
  - subp, 280
  - VEC\_ADD, 280
  - VEC\_SUB, 280
- FFLAS::vectorised::unswitch, 284
  - axpyp, 285
  - modp, 285
  - scalp, 286
- fflas\_101.C, 1107
  - main, 1107
- fflas\_101\_lvl1.C, 1107
  - main, 1107
- FFLAS\_BASE
  - FFLAS, 75
- fflas\_bounds.inl, 827
  - \_\_FFLASFFPACK\_fflas\_bounds\_INL, 827
  - FFLAS\_INT\_TYPE, 827
- fflas\_c.h, 951
  - fadd\_1\_modular\_double, 957
  - fadd\_2\_modular\_double, 960
  - faddin\_1\_modular\_double, 957
  - faddin\_2\_modular\_double, 961
  - fassign\_1\_modular\_double, 955
  - fassign\_2\_modular\_double, 957
  - faxpy\_1\_modular\_double, 956
  - faxpy\_2\_modular\_double, 960
  - fdot\_1\_modular\_double, 956
  - fequal\_1\_modular\_double, 955
  - fequal\_2\_modular\_double, 958
  - FFLAS\_C\_BASE, 954
  - FFLAS\_C\_DIAG, 954
  - FFLAS\_C\_ORDER, 953
  - FFLAS\_C\_SIDE, 954
  - FFLAS\_C\_TRANSPOSE, 953
  - FFLAS\_C\_UPLO, 953
  - FFLAS\_COMPILED, 953
  - FflasColMajor, 953
  - FflasDouble, 954
  - FflasFloat, 954
  - FflasGeneric, 954
  - FflasLeft, 954
  - FflasLower, 954
  - FflasNonUnit, 954
  - FflasNoTrans, 953
  - FflasRight, 954
  - FflasRowMajor, 953
  - FflasTrans, 953
  - FflasUnit, 954
  - FflasUpper, 954
  - fgemm\_3\_modular\_double, 962
  - fgemv\_2\_modular\_double, 961
  - fger\_2\_modular\_double, 961
  - fidentity\_2\_modular\_double, 958
  - fiszero\_1\_modular\_double, 955
  - fiszero\_2\_modular\_double, 958
  - fmove\_2\_modular\_double, 960
  - fneg\_1\_modular\_double, 955
  - fneg\_2\_modular\_double, 959
  - fnegin\_1\_modular\_double, 955
  - fnegin\_2\_modular\_double, 959
  - freduce\_1\_modular\_double, 954
  - freduce\_2\_modular\_double, 959
  - freducein\_1\_modular\_double, 954
  - freducein\_2\_modular\_double, 958
  - fscal\_1\_modular\_double, 956
  - fscal\_2\_modular\_double, 959
  - fscalin\_1\_modular\_double, 956
  - fscalin\_2\_modular\_double, 959
  - fsquare\_3\_modular\_double, 963
  - fsub\_1\_modular\_double, 957
  - fsub\_2\_modular\_double, 960

- fsubin\_1\_modular\_double, 957
- fsubin\_2\_modular\_double, 960
- fswap\_1\_modular\_double, 956
- ftmm\_3\_modular\_double, 962
- ftsm\_3\_modular\_double, 962
- ftsv\_2\_modular\_double, 961
- fzero\_1\_modular\_double, 955
- fzero\_2\_modular\_double, 958
- FFLAS\_C\_BASE
  - fflas\_c.h, 954
- FFLAS\_C\_DIAG
  - fflas\_c.h, 954
  - ffpack\_c.h, 1008
- FFLAS\_C\_ORDER
  - fflas\_c.h, 953
  - ffpack\_c.h, 1008
- FFLAS\_C\_SIDE
  - fflas\_c.h, 954
  - ffpack\_c.h, 1008
- FFLAS\_C\_TRANSPOSE
  - fflas\_c.h, 953
  - ffpack\_c.h, 1008
- FFLAS\_C\_UPLO
  - fflas\_c.h, 953
  - ffpack\_c.h, 1008
- FFLAS\_COMPILED
  - fflas\_c.h, 953
  - ffpack\_inst.C, 1025
  - ffpack\_inst.h, 1026
- fflas\_const\_cast
  - FFPACK, 364, 377
- fflas\_delete
  - FFLAS, 154, 188
- FFLAS\_DIAG
  - FFLAS, 75
- FFLAS\_ELT
  - fflas\_L1\_inst.C, 963, 964
  - fflas\_L1\_inst.h, 964, 965
  - fflas\_L2\_inst.C, 967
  - fflas\_L2\_inst.h, 968
  - fflas\_L3\_inst.C, 970, 971
  - fflas\_L3\_inst.h, 971, 972
  - ffpack\_inst.C, 1025
  - ffpack\_inst.h, 1026
- fflas\_enum.h, 827
- fflas\_fadd.h, 828
- fflas\_fadd.inl, 829
  - \_\_FFLASFFPACK\_fadd\_INL, 830
- fflas\_fassign.h, 830
- fflas\_fassign.inl, 831
  - \_\_FFLASFFPACK\_fassign\_INL, 831
- fflas\_faxpy.inl, 831
  - \_\_FFLASFFPACK\_faxpy\_INL, 832
- fflas\_fdot.inl, 832
  - \_\_FFLASFFPACK\_fdot\_INL, 833
- fflas\_fgemm.inl, 833
  - \_\_FFLASFFPACK\_fgemm\_INL, 835
- fflas\_fgmv.inl, 842
  - \_\_FFLASFFPACK\_fgmv\_INL, 843
- fflas\_fgmv\_mp.inl, 844
  - \_\_FFLASFFPACK\_fgmv\_mp\_INL, 844
- fflas\_fger.inl, 844
  - \_\_FFLASFFPACK\_fger\_INL, 845
- fflas\_fger\_mp.inl, 845
  - \_\_FFPACK\_fger\_mp\_INL, 846
- FFLAS\_FIELD
  - fflas\_L1\_inst.C, 963, 964
  - fflas\_L1\_inst.h, 964, 965
  - fflas\_L2\_inst.C, 967
  - fflas\_L2\_inst.h, 968
  - fflas\_L3\_inst.C, 970, 971
  - fflas\_L3\_inst.h, 971, 972
  - ffpack\_inst.C, 1025
  - ffpack\_inst.h, 1026
- FFLAS\_FORMAT
  - FFLAS, 76
- fflas\_freduce.h, 846
- fflas\_freduce.inl, 847
  - \_\_FFLASFFPACK\_fflas\_freduce\_INL, 849
  - FFLASFFPACK\_COPY\_REDUCE, 849
- fflas\_freduce\_mp.inl, 849
  - \_\_FFLASFFPACK\_fflas\_freduce\_mp\_INL, 849
- fflas\_freivalds.inl, 849
  - \_\_FFLASFFPACK\_freivalds\_INL, 849
- fflas\_fscal.h, 850
- fflas\_fscal.inl, 850
  - \_\_FFLASFFPACK\_fscal\_INL, 851
- fflas\_fscal\_mp.inl, 851
  - \_\_FFLASFFPACK\_fscal\_mp\_INL, 852
- fflas\_fsyr2k.inl, 852
  - \_\_FFLASFFPACK\_fflas\_fsyr2k\_INL, 852
- fflas\_fsyrk.inl, 853
  - \_\_FFLASFFPACK\_fflas\_fsyrk\_INL, 854
- fflas\_fsyrk\_strassen.inl, 854
  - \_\_FFLASFFPACK\_fflas\_fsyrk\_strassen\_INL, 855
- fflas\_ftmm.inl, 855
  - \_\_FFLASFFPACK\_ftmm\_INL, 856
- fflas\_ftsm.inl, 856
  - \_\_FFLASFFPACK\_ftsm\_INL, 856
- fflas\_ftsm\_mp.inl, 857
  - \_\_FFPACK\_ftsm\_mp\_INL, 857
- fflas\_ftsv.inl, 857
  - \_\_FFLASFFPACK\_ftsv\_INL, 858
- fflas\_helpers.inl, 858
  - \_\_FFLASFFPACK\_fflas\_fflas\_mmhelper\_INL, 859
- FFLAS\_INT\_TYPE
  - fflas\_bounds.inl, 827
- fflas\_intrinsic.h, 1044
- fflas\_io.h, 1044
- fflas\_L1\_inst.C, 963
  - \_\_FFLAS\_L1\_INST\_C, 963
- FFLAS\_ELT, 963, 964
- FFLAS\_FIELD, 963, 964
- INST\_OR\_DECL, 963
- fflas\_L1\_inst.h, 964
  - FFLAS\_ELT, 964, 965

- FFLAS\_FIELD, 964, 965
- INST\_OR\_DECL, 964
- fflas\_L1\_inst\_impl.inl, 965
- fflas\_L2\_inst.C, 966
  - \_\_FFLAS\_L2\_INST\_C, 967
  - FFLAS\_ELT, 967
  - FFLAS\_FIELD, 967
  - INST\_OR\_DECL, 967
- fflas\_L2\_inst.h, 967
  - FFLAS\_ELT, 968
  - FFLAS\_FIELD, 968
  - INST\_OR\_DECL, 968
- fflas\_L2\_inst\_impl.inl, 968
- fflas\_L3\_inst.C, 970
  - \_\_FFLAS\_L3\_INST\_C, 970
  - FFLAS\_ELT, 970, 971
  - FFLAS\_FIELD, 970, 971
  - INST\_OR\_DECL, 970
- fflas\_L3\_inst.h, 971
  - FFLAS\_ELT, 971, 972
  - FFLAS\_FIELD, 971, 972
  - INST\_OR\_DECL, 971
- fflas\_L3\_inst\_impl.inl, 972
  - \_\_FFLAS\_\_TRSM\_READONLY, 973
- fflas\_level1.inl, 862
  - \_\_FFLASFFPACK\_fflas\_fflas\_level1\_INL, 864
- fflas\_level2.inl, 865
  - \_\_FFLASFFPACK\_fflas\_fflas\_level2\_INL, 867
- fflas\_level3.inl, 867
  - \_\_FFLASFFPACK\_fflas\_fflas\_level3\_INL, 870
  - \_\_FFLAS\_\_TRSM\_READONLY, 870
- fflas\_lvl1.C, 973
  - fadd\_1\_modular\_double, 976
  - faddin\_1\_modular\_double, 977
  - fassign\_1\_modular\_double, 975
  - faxpy\_1\_modular\_double, 976
  - fdot\_1\_modular\_double, 976
  - fequal\_1\_modular\_double, 975
  - fiszero\_1\_modular\_double, 975
  - fneg\_1\_modular\_double, 974
  - fnegin\_1\_modular\_double, 974
  - freduce\_1\_modular\_double, 974
  - freducein\_1\_modular\_double, 974
  - fscal\_1\_modular\_double, 975
  - fscaln\_1\_modular\_double, 975
  - fsub\_1\_modular\_double, 976
  - fsubin\_1\_modular\_double, 977
  - fswap\_1\_modular\_double, 976
  - fzero\_1\_modular\_double, 974
- fflas\_lvl2.C, 977
  - fadd\_2\_modular\_double, 981
  - faddin\_2\_modular\_double, 981
  - fassign\_2\_modular\_double, 978
  - faxpy\_2\_modular\_double, 980
  - fequal\_2\_modular\_double, 978
  - fgemv\_2\_modular\_double, 981
  - fger\_2\_modular\_double, 982
  - fidentity\_2\_modular\_double, 979
  - fiszero\_2\_modular\_double, 979
  - fmove\_2\_modular\_double, 980
  - fneg\_2\_modular\_double, 980
  - fnegin\_2\_modular\_double, 979
  - freduce\_2\_modular\_double, 979
  - freducein\_2\_modular\_double, 979
  - fscal\_2\_modular\_double, 980
  - fscaln\_2\_modular\_double, 980
  - fsub\_2\_modular\_double, 981
  - fsubin\_2\_modular\_double, 981
  - ftsv\_2\_modular\_double, 982
  - fzero\_2\_modular\_double, 978
- fflas\_lvl3.C, 982
  - fgemm\_3\_modular\_double, 983
  - fsquare\_3\_modular\_double, 984
  - ftmm\_3\_modular\_double, 983
  - ftsm\_3\_modular\_double, 983
- fflas\_memory.h, 1045
- fflas\_new
  - FFLAS, 155, 156, 188
- FFLAS\_ORDER
  - FFLAS, 74
- fflas\_pfgemm.inl, 870
  - \_\_FFLASFFPACK\_DIMKPENALTY, 870
  - \_\_FFLASFFPACK\_SEQPARTHRESHOLD, 870
  - \_\_FFLASFFPACK\_fflas\_pfgemm\_INL, 870
- fflas\_pftsm.inl, 871
  - \_\_FFLASFFPACK\_fflas\_pftsm\_INL, 871
  - PTRSM\_HYBRID\_THRESHOLD, 871
- fflas\_plevel1.h, 1031
- fflas\_randommatrix.h, 1046
- FFLAS\_SIDE
  - FFLAS, 75
- fflas\_simd.h, 871
  - CONST, 872
  - FLOAT\_MOD, 873
  - INLINE, 872
  - NORML\_MOD, 872
  - PURE, 872
  - Simd, 873
  - SIMD\_INT, 872
- fflas\_sparse.C, 984
- fflas\_sparse.h, 881
  - \_\_FFLASFFPACK\_CACHE\_LINE\_SIZE, 885
  - assume\_aligned, 885
  - DENSE\_THRESHOLD, 885
  - index\_t, 885
  - ROUND\_DOWN, 885
- fflas\_sparse.inl, 886
  - \_\_FFLASFFPACK\_fflas\_fflas\_sparse\_INL, 887
- FFLAS\_TRANSPOSE
  - FFLAS, 74
- fflas\_transpose.h, 913
  - FFLAS\_TRANSPOSE\_BLOCKSIZE, 914
  - LD, 914
  - ST, 914
- FFLAS\_TRANSPOSE\_BLOCKSIZE
  - fflas\_transpose.h, 914

- FFLAS\_UPLO
  - FFLAS, [75](#)
- FflasAuto
  - FFLAS, [76](#)
- FflasBinary
  - FFLAS, [76](#)
- FflasColMajor
  - FFLAS, [74](#)
  - fflas\_c.h, [953](#)
  - ffpack\_c.h, [1008](#)
- FflasDense
  - FFLAS, [76](#)
- FflasDouble
  - FFLAS, [76](#)
  - fflas\_c.h, [954](#)
- FFLASFFPACK\_abort
  - debug.h, [1044](#)
- FFLASFFPACK\_check
  - debug.h, [1044](#)
- FFLASFFPACK\_checkers\_fflas\_inl\_H
  - checkers\_fflas.inl, [809](#)
- FFLASFFPACK\_checkers\_ffpack\_inl\_H
  - checkers\_ffpack.inl, [810](#)
- FFLASFFPACK\_COPY\_REDUCE
  - fflas\_freduce.inl, [849](#)
- FFLASFFPACK\_PERM\_BKSIZE
  - ffpack\_permutation.inl, [941](#)
- FflasFloat
  - FFLAS, [76](#)
  - fflas\_c.h, [954](#)
- FflasGeneric
  - FFLAS, [76](#)
  - fflas\_c.h, [954](#)
- FflasLeft
  - FFLAS, [75](#)
  - fflas\_c.h, [954](#)
  - ffpack\_c.h, [1009](#)
- FflasLeftTri
  - FFLAS, [75](#)
- FflasLower
  - FFLAS, [75](#)
  - fflas\_c.h, [954](#)
  - ffpack\_c.h, [1008](#)
- FflasMaple
  - FFLAS, [76](#)
- FflasMath
  - FFLAS, [76](#)
- FflasNonUnit
  - FFLAS, [75](#)
  - fflas\_c.h, [954](#)
  - ffpack\_c.h, [1008](#)
- FflasNoTrans
  - FFLAS, [75](#)
  - fflas\_c.h, [953](#)
  - ffpack\_c.h, [1008](#)
- FflasRight
  - FFLAS, [75](#)
  - fflas\_c.h, [954](#)
- ffpack\_c.h, [1009](#)
- FflasRightTri
  - FFLAS, [75](#)
- FflasRowMajor
  - FFLAS, [74](#)
  - fflas\_c.h, [953](#)
  - ffpack\_c.h, [1008](#)
- FflasSageMath
  - FFLAS, [76](#)
- FflasSMS
  - FFLAS, [76](#)
- FflasTrans
  - FFLAS, [75](#)
  - fflas\_c.h, [953](#)
  - ffpack\_c.h, [1008](#)
- FflasUnit
  - FFLAS, [75](#)
  - fflas\_c.h, [954](#)
  - ffpack\_c.h, [1008](#)
- FflasUpper
  - FFLAS, [75](#)
  - fflas\_c.h, [954](#)
  - ffpack\_c.h, [1008](#)
- FFPACK, [42](#), [287](#)
  - \_PLUQ, [363](#)
  - applyP, [304](#), [305](#), [365](#)
  - applyP\_block, [357](#)
  - Bruhat2EchelonPermutation, [345](#)
  - buildMatrix, [351](#)
  - CharPoly, [323](#), [324](#), [351](#), [369](#), [370](#)
  - Checker\_charpoly, [303](#)
  - Checker\_Det, [303](#)
  - Checker\_invert, [303](#)
  - Checker\_PLUQ, [303](#)
  - chooseField, [390](#)
  - chooseField< Givaro::ZRing< double > >, [391](#)
  - chooseField< Givaro::ZRing< float > >, [390](#)
  - chooseField< Givaro::ZRing< int32\_t > >, [390](#)
  - chooseField< Givaro::ZRing< int64\_t > >, [390](#)
  - ColRankProfileSubmatrix, [335](#), [374](#)
  - ColRankProfileSubmatrixIndices, [334](#), [374](#)
  - ColumnEchelonForm, [316](#), [317](#), [368](#)
  - ColumnRankProfile, [331](#), [332](#), [373](#)
  - composePermutationsLLL, [360](#)
  - composePermutationsLLM, [360](#)
  - composePermutationsMLM, [361](#)
  - CompressToBlockBiDiagonal, [344](#)
  - ComputeRPermutation, [346](#), [348](#)
  - cyclic\_shift\_col, [362](#), [365](#)
  - cyclic\_shift\_mathPerm, [361](#)
  - cyclic\_shift\_row, [361](#), [364](#)
  - cyclic\_shift\_row\_col, [361](#), [364](#)
  - Danilevski, [350](#)
  - Det, [327](#), [328](#), [351](#), [352](#), [371](#), [372](#)
  - doApplyS, [357](#)
  - doApplyT, [358](#)
  - ExpandBlockBiDiagonalToBruhat, [344](#)
  - expandLCRE, [349](#)

- failure, [377](#)
- fflas\_const\_cast, [364](#), [377](#)
- fgesv, [308](#), [309](#), [366](#)
- fgetrs, [306](#), [307](#), [365](#)
- ForceCheck\_charpoly, [304](#)
- ForceCheck\_Det, [304](#)
- ForceCheck\_invert, [304](#)
- ForceCheck\_PLUQ, [303](#)
- fsytrf, [312](#), [313](#)
- fsytrf\_BC\_Crout, [352](#)
- fsytrf\_BC\_RL, [352](#)
- fsytrf\_LOW\_RPM\_BC\_Crout, [353](#)
- fsytrf\_nonunit, [313](#), [353](#), [354](#)
- fsytrf\_RPM, [354](#)
- fsytrf\_UP\_RPM, [353](#)
- fsytrf\_UP\_RPM\_BC\_Crout, [353](#)
- fsytrf\_UP\_RPM\_BC\_RL, [352](#)
- ftssyr2k, [311](#)
- ftstr, [311](#)
- fttri, [309](#), [366](#)
- fttrm, [310](#), [367](#)
- getEchelonForm, [337](#), [338](#)
- getEchelonForm< FFLAS\_FIELD< FFLAS\_ELT >  
>, [375](#)
- getEchelonTransform, [338](#)
- getEchelonTransform< FFLAS\_FIELD< FFLAS\_ELT  
> >, [375](#)
- getLTBruhatGen, [342](#), [343](#)
- getReducedEchelonForm, [339](#), [340](#)
- getReducedEchelonForm< FFLAS\_FIELD<  
FFLAS\_ELT > >, [376](#)
- getReducedEchelonTransform, [341](#)
- getReducedEchelonTransform< FFLAS\_FIELD<  
FFLAS\_ELT > >, [376](#)
- getTriangular, [336](#)
- getTriangular< FFLAS\_FIELD< FFLAS\_ELT > >,  
[374](#), [375](#)
- getTridiagonal, [354](#)
- Invert, [321](#), [369](#)
- Invert2, [322](#), [369](#)
- isOdd, [377](#)
- IsSingular, [326](#), [371](#)
- KrylovElim, [371](#)
- LAPACKPerm2MathPerm, [304](#)
- LeadingSubmatrixRankProfiles, [332](#)
- LQUPtoInverseOfFullRankMinor, [346](#), [377](#)
- LTBruhatGen, [342](#)
- LTQSorter, [343](#)
- LUdivine, [315](#), [355](#), [367](#)
- LUdivine\_gauss, [354](#), [368](#)
- LUdivine\_small, [355](#), [367](#)
- MathPerm2LAPACKPerm, [304](#)
- MatrixApplyS, [357](#), [358](#)
- MatrixApplyT, [359](#)
- MatVecMinPoly, [325](#), [370](#)
- MinPoly, [324](#), [370](#)
- MonotonicApplyP, [306](#)
- MonotonicCompress, [356](#)
- MonotonicCompressCycles, [356](#)
- MonotonicCompressMorePivots, [356](#)
- MonotonicExpand, [357](#)
- NonZeroRandomMatrix, [377](#), [378](#)
- NullSpaceBasis, [329](#), [373](#)
- pColumnEchelonForm, [316](#)
- pColumnRankProfile, [332](#)
- pDet, [327](#)
- PermApplyS, [358](#)
- PermApplyT, [360](#)
- PLUQ, [313](#)–[315](#), [363](#), [364](#), [367](#)
- PLUQ\_basecaseCrout, [362](#)
- PLUQ\_basecaseV2, [362](#)
- PLUQ\_basecaseV3, [362](#)
- PLUQtoEchelonPermutation, [341](#)
- pPLUQ, [314](#)
- pRank, [326](#)
- pReducedColumnEchelonForm, [319](#)
- pReducedRowEchelonForm, [320](#)
- productBruhatxTS, [346](#), [349](#)
- pRowEchelonForm, [318](#)
- pRowRankProfile, [331](#)
- pSolve, [329](#)
- RandInt, [381](#)
- RandomIndexSubset, [382](#)
- RandomLTQSMatrixWithRankandQSorter, [390](#)
- RandomLTQSRankProfileMatrix, [384](#)
- RandomMatrix, [378](#), [379](#)
- RandomMatrixWithDet, [389](#)
- RandomMatrixWithRank, [381](#), [382](#)
- RandomMatrixWithRankandRandomRPM, [387](#)
- RandomMatrixWithRankandRPM, [384](#), [385](#)
- RandomNullSpaceVector, [329](#), [347](#), [373](#)
- RandomPermutation, [383](#)
- RandomRankProfileMatrix, [383](#)
- RandomSymmetricMatrix, [381](#)
- RandomSymmetricMatrixWithRankandRandom-  
RPM, [388](#)
- RandomSymmetricMatrixWithRankandRPM, [385](#),  
[386](#)
- RandomSymmetricRankProfileMatrix, [384](#)
- RandomTriangularMatrix, [379](#), [380](#)
- Rank, [325](#), [326](#), [371](#)
- RankProfileFromLU, [332](#)
- ReducedColumnEchelonForm, [318](#), [319](#), [369](#)
- ReducedRowEchelonForm, [319](#), [320](#), [368](#)
- RowEchelonForm, [317](#), [318](#), [368](#)
- RowRankProfile, [330](#), [331](#), [373](#)
- RowRankProfileSubmatrix, [334](#), [374](#)
- RowRankProfileSubmatrixIndices, [333](#), [374](#)
- Solve, [328](#), [372](#)
- solveLB, [348](#), [372](#)
- solveLB2, [348](#), [372](#)
- SpecRankProfile, [371](#)
- swapval, [383](#)
- threads\_fgemm, [363](#)
- threads\_ftsm, [363](#)
- TInverter, [345](#), [348](#)

- trinv\_left, [310](#), [367](#)
- ffpack-fgesv.C, [1107](#)
  - main, [1108](#)
- ffpack-solve.C, [1108](#)
  - main, [1108](#)
- ffpack.C, [984](#)
  - applyP\_modular\_double, [990](#)
  - ColRankProfileSubmatrix\_modular\_double, [1002](#)
  - ColRankProfileSubmatrixIndices\_modular\_double, [1002](#)
  - ColumnEchelonForm\_modular\_double, [992](#)
  - ColumnEchelonForm\_modular\_float, [993](#)
  - ColumnEchelonForm\_modular\_int32\_t, [994](#)
  - ColumnRankProfile\_modular\_double, [1001](#)
  - composePermutationsLLL, [989](#)
  - composePermutationsLLM, [989](#)
  - composePermutationsMLM, [989](#)
  - cyclic\_shift\_col\_modular\_double, [990](#)
  - cyclic\_shift\_mathPerm, [989](#)
  - cyclic\_shift\_row\_modular\_double, [989](#)
  - Det\_modular\_double, [999](#)
  - fgesv\_modular\_double, [991](#)
  - fgesvin\_modular\_double, [991](#)
  - fgetrsin\_modular\_double, [990](#)
  - fgetrsv\_modular\_double, [990](#)
  - fttrtri\_modular\_double, [991](#)
  - fttrtm\_modular\_double, [992](#)
  - getEchelonForm\_modular\_double, [1003](#)
  - getEchelonFormin\_modular\_double, [1003](#)
  - getEchelonTransform\_modular\_double, [1003](#)
  - getReducedEchelonForm\_modular\_double, [1004](#)
  - getReducedEchelonFormin\_modular\_double, [1004](#)
  - getReducedEchelonTransform\_modular\_double, [1004](#)
  - getTriangular\_modular\_double, [1002](#)
  - getTriangularin\_modular\_double, [1003](#)
  - Invert2\_modular\_double, [998](#)
  - Invert\_modular\_double, [998](#)
  - Invertin\_modular\_double, [998](#)
  - IsSingular\_modular\_double, [999](#)
  - KrylovElim\_modular\_double, [998](#)
  - LAPACKPerm2MathPerm, [988](#)
  - LeadingSubmatrixRankProfiles, [1001](#)
  - LUdivine\_modular\_double, [992](#)
  - MathPerm2LAPACKPerm, [988](#)
  - MatrixApplyS\_modular\_double, [988](#)
  - MatrixApplyT\_modular\_double, [988](#)
  - NullSpaceBasis\_modular\_double, [1000](#)
  - pColumnEchelonForm\_modular\_double, [995](#)
  - pColumnEchelonForm\_modular\_float, [996](#)
  - pColumnEchelonForm\_modular\_int32\_t, [997](#)
  - PermApplyS\_double, [988](#)
  - PermApplyT\_double, [989](#)
  - PLUQ\_modular\_double, [992](#)
  - PLUQtoEchelonPermutation, [1004](#)
  - pReducedColumnEchelonForm\_modular\_double, [996](#)
  - pReducedColumnEchelonForm\_modular\_float, [996](#)
  - pReducedColumnEchelonForm\_modular\_int32\_t, [997](#)
  - pReducedRowEchelonForm\_modular\_double, [996](#)
  - pReducedRowEchelonForm\_modular\_float, [997](#)
  - pReducedRowEchelonForm\_modular\_int32\_t, [998](#)
  - pRowEchelonForm\_modular\_double, [995](#)
  - pRowEchelonForm\_modular\_float, [996](#)
  - pRowEchelonForm\_modular\_int32\_t, [997](#)
  - RandomNullSpaceVector\_modular\_double, [1000](#)
  - Rank\_modular\_double, [999](#)
  - RankProfileFromLU, [1001](#)
  - ReducedColumnEchelonForm\_modular\_double, [993](#)
  - ReducedColumnEchelonForm\_modular\_float, [994](#)
  - ReducedColumnEchelonForm\_modular\_int32\_t, [995](#)
  - ReducedRowEchelonForm\_modular\_double, [993](#)
  - ReducedRowEchelonForm\_modular\_float, [994](#)
  - ReducedRowEchelonForm\_modular\_int32\_t, [995](#)
  - RowEchelonForm\_modular\_double, [992](#)
  - RowEchelonForm\_modular\_float, [993](#)
  - RowEchelonForm\_modular\_int32\_t, [994](#)
  - RowRankProfile\_modular\_double, [1001](#)
  - RowRankProfileSubmatrix\_modular\_double, [1002](#)
  - RowRankProfileSubmatrixIndices\_modular\_double, [1001](#)
  - Solve\_modular\_double, [999](#)
  - solveLB2\_modular\_double, [1000](#)
  - solveLB\_modular\_double, [1000](#)
  - SpecRankProfile\_modular\_double, [999](#)
  - trinv\_left\_modular\_double, [991](#)
  - ffpack.doxy, [914](#)
  - ffpack.h, [914](#)
    - \_\_FFLASFFPACK\_FTRSSYR2K\_THRESHOLD, [923](#)
    - \_\_FFLASFFPACK\_FTRSTR\_THRESHOLD, [923](#)
  - ffpack.inl, [924](#)
    - \_\_FFLASFFPACK\_ffpack\_INL, [925](#)
  - FFPACK::Protected, [391](#)
    - ArithProg, [395](#)
    - CompressRows, [396](#)
    - CompressRowsQA, [397](#)
    - CompressRowsQK, [397](#)
    - Danilevski, [395](#)
    - DeCompressRows, [397](#)
    - DeCompressRowsQA, [398](#)
    - DeCompressRowsQK, [397](#)
    - fgemv\_kgf, [394](#)
    - GaussJordan, [393](#)
    - Hybrid\_KGF\_LUK\_MinPoly, [396](#)
    - KellerGehrig, [393](#)
    - KGFast, [394](#)
    - KGFast\_generalized, [394](#)
    - LUdivine\_construct, [392](#), [398](#)
    - LUKrylov, [394](#)
    - LUKrylov\_KGFast, [395](#)



- MatVecMinPoly, 396
- newD, 396
- RandomKrylovPrecond, 395
- updatedD, 396
- ffpack\_bruhatgen.inl, 925
- \_\_FFLASFFPACK\_ffpack\_bruhatgen\_inl, 926
- ffpack\_c.h, 1005
- applyP\_modular\_double, 1011
- ColRankProfileSubmatrix\_modular\_double, 1022
- ColRankProfileSubmatrixIndices\_modular\_double, 1021
- ColumnEchelonForm\_modular\_double, 1014
- ColumnEchelonForm\_modular\_float, 1015
- ColumnEchelonForm\_modular\_int32\_t, 1015
- ColumnRankProfile\_modular\_double, 1020
- composePermutationsLLL, 1011
- composePermutationsLLM, 1010
- composePermutationsMLM, 1011
- cyclic\_shift\_col\_modular\_double, 1011
- cyclic\_shift\_mathPerm, 1011
- cyclic\_shift\_row\_modular\_double, 1011
- Det\_modular\_double, 1019
- FFLAS\_C\_DIAG, 1008
- FFLAS\_C\_ORDER, 1008
- FFLAS\_C\_SIDE, 1008
- FFLAS\_C\_TRANSPOSE, 1008
- FFLAS\_C\_UPLO, 1008
- FflasColMajor, 1008
- FflasLeft, 1009
- FflasLower, 1008
- FflasNonUnit, 1008
- FflasNoTrans, 1008
- FflasRight, 1009
- FflasRowMajor, 1008
- FflasTrans, 1008
- FflasUnit, 1008
- FflasUpper, 1008
- FFPACK\_C\_CHARPOLY\_TAG, 1009
- FFPACK\_C\_LU\_TAG, 1009
- FFPACK\_C\_MINPOLY\_TAG, 1009
- FFPACK\_COMPILED, 1008
- FfpackArithProg, 1009
- FfpackDanilevski, 1009
- FfpackDense, 1009
- FfpackHybrid, 1009
- FfpackKG, 1009
- FfpackKGF, 1009
- FfpackKGFast, 1009
- FfpackKGFastG, 1009
- FfpackLUK, 1009
- FfpackSingular, 1009
- FfpackSlabRecursive, 1009
- FfpackTileRecursive, 1009
- fgesv\_modular\_double, 1012
- fgesvin\_modular\_double, 1012
- fgetrs\_modular\_double, 1012
- fgetrsin\_modular\_double, 1012
- ftrtri\_modular\_double, 1013
- ftrtrm\_modular\_double, 1013
- getEchelonForm\_modular\_double, 1022
- getEchelonFormin\_modular\_double, 1023
- getEchelonTransform\_modular\_double, 1023
- getReducedEchelonForm\_modular\_double, 1023
- getReducedEchelonFormin\_modular\_double, 1024
- getReducedEchelonTransform\_modular\_double, 1024
- getTriangular\_modular\_double, 1022
- getTriangularin\_modular\_double, 1022
- Invert2\_modular\_double, 1018
- Invert\_modular\_double, 1018
- Invertin\_modular\_double, 1018
- IsSingular\_modular\_double, 1019
- KrylovElim\_modular\_double, 1018
- LAPACKPerm2MathPerm, 1009
- LeadingSubmatrixRankProfiles, 1021
- LUdivine\_gauss\_modular\_double, 1014
- LUdivine\_modular\_double, 1013
- LUdivine\_small\_modular\_double, 1014
- MathPerm2LAPACKPerm, 1009
- MatrixApplyS\_modular\_double, 1010
- MatrixApplyT\_modular\_double, 1010
- NullSpaceBasis\_modular\_double, 1020
- PermApplyS\_double, 1010
- PermApplyT\_double, 1010
- PLUQ\_modular\_double, 1013
- PLUQtoEchelonPermutation, 1024
- RandomNullSpaceVector\_modular\_double, 1020
- Rank\_modular\_double, 1019
- RankProfileFromLU, 1021
- ReducedColumnEchelonForm\_modular\_double, 1016
- ReducedColumnEchelonForm\_modular\_float, 1016
- ReducedColumnEchelonForm\_modular\_int32\_t, 1017
- ReducedRowEchelonForm2\_modular\_double, 1017
- ReducedRowEchelonForm\_modular\_double, 1016
- ReducedRowEchelonForm\_modular\_float, 1016
- ReducedRowEchelonForm\_modular\_int32\_t, 1017
- REF\_modular\_double, 1017
- RowEchelonForm\_modular\_double, 1014
- RowEchelonForm\_modular\_float, 1015
- RowEchelonForm\_modular\_int32\_t, 1015
- RowRankProfile\_modular\_double, 1020
- RowRankProfileSubmatrix\_modular\_double, 1021
- RowRankProfileSubmatrixIndices\_modular\_double, 1021
- Solve\_modular\_double, 1019
- solveLB2\_modular\_double, 1020
- solveLB\_modular\_double, 1019
- SpecRankProfile\_modular\_double, 1018
- trinv\_left\_modular\_double, 1013
- FFPACK\_C\_CHARPOLY\_TAG
- ffpack\_c.h, 1009

FFPACK\_C\_LU\_TAG  
     ffpack\_c.h, 1009  
 FFPACK\_C\_MINPOLY\_TAG  
     ffpack\_c.h, 1009  
 ffpack\_charpoly.inl, 926  
     \_\_FFLASFFPACK\_charpoly\_INL, 927  
 ffpack\_charpoly\_danilevski.inl, 927  
     \_\_FFLASFFPACK\_ffpack\_charpoly\_danilveski\_INL, 927  
 ffpack\_charpoly\_kgfast.inl, 927  
     \_\_FFLASFFPACK\_ffpack\_charpoly\_kgfast\_INL, 928  
 ffpack\_charpoly\_kgfastgeneralized.inl, 928  
     \_\_FFLASFFPACK\_ffpack\_charpoly\_kgfastgeneralized\_INL, 928  
     FFPACK\_pluq\_mp\_INL, 942  
 ffpack\_charpoly\_kglu.inl, 928  
     \_\_FFLASFFPACK\_ffpack\_charpoly\_kglu\_INL, 929  
 ffpack\_charpoly\_mp.inl, 929  
     \_\_FFPACK\_charpoly\_mp\_INL, 929  
 FFPACK\_COMPILED  
     ffpack\_c.h, 1008  
 ffpack\_det\_mp.inl, 929  
     \_\_FFPACK\_det\_mp\_INL, 930  
 ffpack\_echelonforms.inl, 930  
     \_\_FFLASFFPACK\_GAUSSJORDAN\_BASECASE, 931  
     \_\_FFLASFFPACK\_ffpack\_echelon\_forms\_INL, 931  
 ffpack\_fgesv.inl, 931  
     \_\_FFLASFFPACK\_ffpack\_fgesv\_INL, 932  
 ffpack\_fgetrs.inl, 932  
     \_\_FFLASFFPACK\_ffpack\_fgetrs\_INL, 932  
 ffpack\_frobenius.inl, 932  
 ffpack\_fsytrf.inl, 933  
     \_\_FFLASFFPACK\_ffpack\_fsytrf\_INL, 934  
 ffpack\_ftrssyr2k.inl, 934  
     \_\_FFLASFFPACK\_ffpack\_ftrssyr2k\_INL, 935  
 ffpack\_ftrstr.inl, 935  
     \_\_FFLASFFPACK\_ffpack\_ftrstr\_INL, 935  
 ffpack\_ftrtr.inl, 935  
     \_\_FFLASFFPACK\_ffpack\_ftrtr\_INL, 936  
     ENABLE\_ALL\_CHECKINGS, 936  
 ffpack\_inst.C, 1024  
     \_\_FFPACK\_INST\_C, 1025  
     FFLAS\_COMPILED, 1025  
     FFLAS\_ELT, 1025  
     FFLAS\_FIELD, 1025  
     INST\_OR\_DECL, 1025  
 ffpack\_inst.h, 1025  
     FFLAS\_COMPILED, 1026  
     FFLAS\_ELT, 1026  
     FFLAS\_FIELD, 1026  
     INST\_OR\_DECL, 1026  
 ffpack\_inst\_implem.inl, 1027  
 ffpack\_invert.inl, 936  
     \_\_FFLASFFPACK\_ffpack\_invert\_INL, 936  
 ffpack\_krylovelim.inl, 937  
     \_\_FFLASFFPACK\_ffpack\_krylovelim\_INL, 937  
 ffpack\_ludivine.inl, 937  
     \_\_FFLASFFPACK\_ffpack\_ludivine\_INL, 937  
 ffpack\_ludivine\_mp.inl, 938  
     \_\_FFPACK\_ludivine\_mp\_INL, 938  
 ffpack\_minpoly.inl, 938  
     \_\_FFLASFFPACK\_ffpack\_minpoly\_INL, 939  
 ffpack\_permutation.inl, 939  
     \_\_FFLASFFPACK\_ffpack\_permutation\_INL, 941  
     FFLASFFPACK\_PERM\_BKSIZE, 941  
 ffpack\_pluq.inl, 941  
     \_\_FFLASFFPACK\_ffpack\_pluq\_INL, 942  
     CROUT, 942  
 ffpack\_pluq\_mp.inl, 942  
 ffpack\_ppluq.inl, 942  
     \_\_FFLASFFPACK\_ffpack\_ppluq\_INL, 943  
     \_\_FFLAS\_\_TRSM\_READONLY, 943  
     PBASECASE\_K, 943  
 ffpack\_rankprofiles.inl, 943  
     \_\_FFLASFFPACK\_ffpack\_rank\_profiles\_INL, 944  
 FfpackArithProg  
     ffpack\_c.h, 1009  
 FfpackDanilevski  
     ffpack\_c.h, 1009  
 FfpackDense  
     ffpack\_c.h, 1009  
 FfpackHybrid  
     ffpack\_c.h, 1009  
 FfpackKG  
     ffpack\_c.h, 1009  
 FfpackKGF  
     ffpack\_c.h, 1009  
 FfpackKGFast  
     ffpack\_c.h, 1009  
 FfpackKGFastG  
     ffpack\_c.h, 1009  
 FfpackLUK  
     ffpack\_c.h, 1009  
 FfpackSingular  
     ffpack\_c.h, 1009  
 FfpackSlabRecursive  
     ffpack\_c.h, 1009  
 FfpackTileRecursive  
     ffpack\_c.h, 1009  
 fgemm  
     FFLAS, 88–90, 92–97, 144, 176–178  
 fgemm\_3\_modular\_double  
     fflas\_c.h, 962  
     fflas\_lvl3.C, 983  
 fgemm\_classical.inl, 835  
 fgemm\_classical\_mp.inl, 835  
     \_\_FFPACK\_fgemm\_classical\_INL, 837  
 fgemm\_convert  
     FFLAS::Protected, 213  
 fgemm\_winograd.inl, 837  
     \_\_FFLASFFPACK\_fflas\_fflas\_fgemm\_winograd\_INL, 839  
     NEWWINO, 839



- fgemv
  - FFLAS, [97–102](#), [173](#), [184](#)
- fgemv\_2\_modular\_double
  - fflas\_c.h, [961](#)
  - fflas\_lvl2.C, [981](#)
- fgemv\_convert
  - FFLAS::Protected, [218](#)
- fgemv\_kgf
  - FFPACK::Protected, [394](#)
- fger
  - FFLAS, [103–106](#), [173](#)
- fger\_2\_modular\_double
  - fflas\_c.h, [961](#)
  - fflas\_lvl2.C, [982](#)
- fger\_convert
  - FFLAS::Protected, [218](#)
- fgesv
  - FFPACK, [308](#), [309](#), [366](#)
- fgesv\_modular\_double
  - ffpack.C, [991](#)
  - ffpack\_c.h, [1012](#)
- fgesvin\_modular\_double
  - ffpack.C, [991](#)
  - ffpack\_c.h, [1012](#)
- fgetrs
  - FFPACK, [306](#), [307](#), [365](#)
- fgetrs\_modular\_double
  - ffpack\_c.h, [1012](#)
- fgetrsin\_modular\_double
  - ffpack.C, [990](#)
  - ffpack\_c.h, [1012](#)
- fgetrsv\_modular\_double
  - ffpack.C, [990](#)
- fidentity
  - FFLAS, [136](#), [137](#), [166](#)
- fidentity\_2\_modular\_double
  - fflas\_c.h, [958](#)
  - fflas\_lvl2.C, [979](#)
- Field
  - Bench< Elt >, [407](#)
  - benchmark-fgemm-rns.C, [788](#)
  - benchmark-pluq.C, [801](#)
  - FieldSimd< \_Field >, [447](#)
  - Sparse< \_Field, SparseMatrix\_t::COO >, [736](#)
  - Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, [737](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR >, [739](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, [741](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, [743](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL >, [744](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, [749](#)
  - Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, [751](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL >, [752](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, [754](#)
  - Test< Elt >, [762](#)
  - test-compressQ.C, [1056](#)
- field
  - associatedDelayedField< const FFPACK::RNSIntegerMod< RNS > >, [404](#)
  - associatedDelayedField< const Givaro::Modular< T, X > >, [405](#)
  - associatedDelayedField< const Givaro::ModularBalanced< T > >, [405](#)
  - associatedDelayedField< const Givaro::ZRing< T > >, [406](#)
  - associatedDelayedField< Field >, [404](#)
  - field-traits.h, [944](#)
  - field.doxy, [947](#)
  - FieldMax
    - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [502](#)
  - FieldMin
    - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [502](#)
  - FieldSimd
    - FieldSimd< \_Field >, [448](#)
  - FieldSimd< \_Field >, [446](#)
    - add, [448](#)
    - add\_r, [449](#)
    - addin, [449](#)
    - addin\_r, [449](#)
    - alignment, [452](#)
    - axpy, [451](#)
    - axpy\_r, [451](#)
    - axpyin, [451](#)
    - axpyin\_r, [451](#)
    - Element, [447](#)
    - Field, [447](#)
    - FieldSimd, [448](#)
    - init, [448](#)
    - maxpy, [452](#)
    - maxpyin, [452](#)
    - mod, [450](#)
    - mul, [450](#)
    - mul\_r, [450](#), [451](#)
    - mulin, [450](#)
    - operator=, [448](#)
    - scalar\_t, [447](#)
    - simd, [447](#)
    - sub, [449](#)
    - sub\_r, [449](#), [450](#)
    - subin, [449](#)
    - subin\_r, [450](#)
    - vect\_size, [452](#)
    - vect\_t, [447](#)
    - zero, [450](#)
  - FieldTraits< FFPACK::RNSInteger< T > >, [453](#)
    - balanced, [453](#)
    - category, [453](#)
  - FieldTraits< FFPACK::RNSIntegerMod< T > >, [453](#)
    - balanced, [454](#)
    - category, [454](#)
  - FieldTraits< Field >, [452](#)
    - balanced, [453](#)
    - category, [453](#)

- FieldTraits< Givaro::Modular< Element > >, [454](#)
  - balanced, [454](#)
  - category, [454](#)
- FieldTraits< Givaro::ModularBalanced< Element > >, [454](#)
  - balanced, [455](#)
  - category, [455](#)
- FieldTraits< Givaro::ZRing< double > >, [455](#)
  - balanced, [455](#)
  - category, [455](#)
- FieldTraits< Givaro::ZRing< float > >, [455](#)
  - balanced, [456](#)
  - category, [456](#)
- FieldTraits< Givaro::ZRing< Givaro::Integer > >, [456](#)
  - balanced, [456](#)
  - category, [456](#)
- FieldTraits< Givaro::ZRing< int16\_t > >, [456](#)
  - balanced, [457](#)
  - category, [457](#)
- FieldTraits< Givaro::ZRing< int32\_t > >, [457](#)
  - balanced, [457](#)
  - category, [457](#)
- FieldTraits< Givaro::ZRing< int64\_t > >, [457](#)
  - balanced, [458](#)
  - category, [457](#)
- FieldTraits< Givaro::ZRing< Reclnt::ruint< K > > >, [458](#)
  - balanced, [458](#)
  - category, [458](#)
- FieldTraits< Givaro::ZRing< uint16\_t > >, [458](#)
  - balanced, [459](#)
  - category, [458](#)
- FieldTraits< Givaro::ZRing< uint32\_t > >, [459](#)
  - balanced, [459](#)
  - category, [459](#)
- FieldTraits< Givaro::ZRing< uint64\_t > >, [459](#)
  - balanced, [459](#)
  - category, [459](#)
- fill\_value
  - benchmark-fgemv.C, [792](#)
- findArgument
  - args-parser.h, [1042](#)
- finit
  - FFLAS, [108](#), [110](#), [129](#), [137](#), [157](#), [167](#)
- finit\_rns
  - FFLAS, [155](#), [156](#)
- finit\_trans\_rns
  - FFLAS, [155](#)
- first\_component
  - Compose< H1, H2 >, [424](#)
- firstBlockSize
  - ForStrategy1D< blocksize\_t, Cut, Param >, [462](#)
- fiszero
  - FFLAS, [131](#), [136](#), [159](#), [166](#)
- fiszero\_1\_modular\_double
  - fflas\_c.h, [955](#)
  - fflas\_lvl1.C, [975](#)
- fiszero\_2\_modular\_double
  - fflas\_c.h, [958](#)
  - fflas\_lvl2.C, [979](#)
- Fixed, [460](#)
- FixedPrecIntTag, [460](#)
- flimits.h, [1048](#)
  - in\_range, [1048](#), [1049](#)
- FLOAT\_MOD
  - fflas\_simd.h, [873](#)
- FloatingPointTestDistribution
  - ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >::FloatingPointTestDistribution, [460](#)
- Floats
  - benchmark-dgemm.C, [782](#)
- floor
  - ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >, [559](#)
  - Simd256\_impl< true, false, true, 8 >, [630](#)
  - Simd512\_impl< true, false, true, 8 >, [712](#)
- fma
  - ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >, [560](#)
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, [561](#)
- fmadd
  - ScalFunctions< Element >, [556](#)
  - Simd128\_impl< true, true, false, 2 >, [572](#)
  - Simd128\_impl< true, true, false, 4 >, [582](#)
  - Simd128\_impl< true, true, false, 8 >, [592](#)
  - Simd128\_impl< true, true, true, 2 >, [600](#)
  - Simd128\_impl< true, true, true, 4 >, [609](#)
  - Simd128\_impl< true, true, true, 8 >, [618](#)
  - Simd256\_impl< true, false, true, 8 >, [629](#)
  - Simd256\_impl< true, true, false, 2 >, [640](#)
  - Simd256\_impl< true, true, false, 4 >, [656](#), [657](#)
  - Simd256\_impl< true, true, false, 8 >, [668](#)
  - Simd256\_impl< true, true, true, 2 >, [676](#)
  - Simd256\_impl< true, true, true, 4 >, [686](#), [692](#)
  - Simd256\_impl< true, true, true, 8 >, [701](#)
  - Simd512\_impl< true, false, true, 8 >, [710](#)
  - Simd512\_impl< true, true, false, 8 >, [721](#)
  - Simd512\_impl< true, true, true, 8 >, [730](#)
- fmaddin
  - ScalFunctions< Element >, [556](#)
  - Simd128\_impl< true, true, false, 2 >, [572](#)
  - Simd128\_impl< true, true, false, 4 >, [582](#)
  - Simd128\_impl< true, true, false, 8 >, [592](#)
  - Simd128\_impl< true, true, true, 2 >, [600](#)
  - Simd128\_impl< true, true, true, 4 >, [609](#)
  - Simd128\_impl< true, true, true, 8 >, [618](#)
  - Simd256\_impl< true, false, true, 8 >, [629](#)
  - Simd256\_impl< true, true, false, 2 >, [640](#)
  - Simd256\_impl< true, true, false, 4 >, [657](#)
  - Simd256\_impl< true, true, false, 8 >, [668](#)
  - Simd256\_impl< true, true, true, 2 >, [676](#)
  - Simd256\_impl< true, true, true, 4 >, [686](#), [692](#)

- Simd256\_impl< true, true, true, 8 >, [701](#)
- Simd512\_impl< true, false, true, 8 >, [711](#)
- Simd512\_impl< true, true, false, 8 >, [721](#)
- Simd512\_impl< true, true, true, 8 >, [730](#)
- fmaddx
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, [561](#)
  - Simd128\_impl< true, true, false, 2 >, [568](#)
  - Simd128\_impl< true, true, false, 4 >, [578](#)
  - Simd128\_impl< true, true, false, 8 >, [588](#)
  - Simd128\_impl< true, true, true, 2 >, [600](#)
  - Simd128\_impl< true, true, true, 4 >, [609](#)
  - Simd128\_impl< true, true, true, 8 >, [618](#)
  - Simd256\_impl< true, true, false, 2 >, [635](#)
  - Simd256\_impl< true, true, false, 4 >, [647](#), [649](#)
  - Simd256\_impl< true, true, false, 8 >, [664](#)
  - Simd256\_impl< true, true, true, 2 >, [676](#)
  - Simd256\_impl< true, true, true, 4 >, [686](#), [692](#)
  - Simd256\_impl< true, true, true, 8 >, [702](#)
  - Simd512\_impl< true, true, false, 8 >, [717](#)
  - Simd512\_impl< true, true, true, 8 >, [730](#)
- fmaddxin
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, [561](#)
  - Simd128\_impl< true, true, false, 2 >, [568](#)
  - Simd128\_impl< true, true, false, 4 >, [578](#)
  - Simd128\_impl< true, true, false, 8 >, [588](#)
  - Simd128\_impl< true, true, true, 2 >, [601](#)
  - Simd128\_impl< true, true, true, 4 >, [609](#)
  - Simd128\_impl< true, true, true, 8 >, [619](#)
  - Simd256\_impl< true, true, false, 2 >, [636](#)
  - Simd256\_impl< true, true, false, 4 >, [647](#), [650](#)
  - Simd256\_impl< true, true, false, 8 >, [664](#)
  - Simd256\_impl< true, true, true, 2 >, [676](#)
  - Simd256\_impl< true, true, true, 4 >, [686](#), [693](#)
  - Simd256\_impl< true, true, true, 8 >, [702](#)
  - Simd512\_impl< true, true, false, 8 >, [717](#)
  - Simd512\_impl< true, true, true, 8 >, [730](#)
- fmove
  - FFLAS, [139](#), [170](#)
- fmove\_2\_modular\_double
  - fflas\_c.h, [960](#)
  - fflas\_lvl2.C, [980](#)
- fmsub
  - ScalFunctions< Element >, [556](#)
  - Simd128\_impl< true, true, false, 2 >, [573](#)
  - Simd128\_impl< true, true, false, 4 >, [583](#)
  - Simd128\_impl< true, true, false, 8 >, [593](#)
  - Simd128\_impl< true, true, true, 2 >, [601](#)
  - Simd128\_impl< true, true, true, 4 >, [610](#)
  - Simd128\_impl< true, true, true, 8 >, [619](#)
  - Simd256\_impl< true, false, true, 8 >, [629](#)
  - Simd256\_impl< true, true, false, 2 >, [640](#)
  - Simd256\_impl< true, true, false, 4 >, [657](#), [658](#)
  - Simd256\_impl< true, true, false, 8 >, [668](#)
  - Simd256\_impl< true, true, true, 2 >, [677](#)
  - Simd256\_impl< true, true, true, 4 >, [687](#), [693](#)
  - Simd256\_impl< true, true, true, 8 >, [702](#)
- fmsubx
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, [562](#)
  - Simd128\_impl< true, true, false, 2 >, [569](#)
  - Simd128\_impl< true, true, false, 4 >, [579](#)
  - Simd128\_impl< true, true, false, 8 >, [589](#)
  - Simd128\_impl< true, true, true, 2 >, [601](#)
  - Simd128\_impl< true, true, true, 4 >, [610](#)
  - Simd128\_impl< true, true, true, 8 >, [619](#)
  - Simd256\_impl< true, true, false, 2 >, [636](#)
  - Simd256\_impl< true, true, false, 4 >, [647](#), [650](#)
  - Simd256\_impl< true, true, false, 8 >, [664](#)
  - Simd256\_impl< true, true, true, 2 >, [677](#)
  - Simd256\_impl< true, true, true, 4 >, [687](#), [693](#)
  - Simd256\_impl< true, true, true, 8 >, [703](#)
  - Simd512\_impl< true, true, false, 8 >, [717](#)
  - Simd512\_impl< true, true, true, 8 >, [731](#)
- fmsubxin
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, [562](#)
  - Simd128\_impl< true, true, false, 2 >, [569](#)
  - Simd128\_impl< true, true, false, 4 >, [579](#)
  - Simd128\_impl< true, true, false, 8 >, [589](#)
  - Simd128\_impl< true, true, true, 2 >, [602](#)
  - Simd128\_impl< true, true, true, 4 >, [610](#)
  - Simd128\_impl< true, true, true, 8 >, [620](#)
  - Simd256\_impl< true, true, false, 2 >, [636](#)
  - Simd256\_impl< true, true, false, 4 >, [648](#), [650](#)
  - Simd256\_impl< true, true, false, 8 >, [664](#)
  - Simd256\_impl< true, true, true, 2 >, [677](#)
  - Simd256\_impl< true, true, true, 4 >, [687](#), [694](#)
  - Simd256\_impl< true, true, true, 8 >, [703](#)
  - Simd512\_impl< true, true, false, 8 >, [718](#)
  - Simd512\_impl< true, true, true, 8 >, [731](#)
- fneg
  - FFLAS, [130](#), [138](#), [159](#), [168](#)
- fneg\_1\_modular\_double

- fflas\_c.h, [955](#)
- fflas\_lvl1.C, [974](#)
- fneg\_2\_modular\_double
  - fflas\_c.h, [959](#)
  - fflas\_lvl2.C, [980](#)
- fnegin
  - FFLAS, [130](#), [138](#), [158](#), [168](#)
- fnegin\_1\_modular\_double
  - fflas\_c.h, [955](#)
  - fflas\_lvl1.C, [974](#)
- fnegin\_2\_modular\_double
  - fflas\_c.h, [959](#)
  - fflas\_lvl2.C, [979](#)
- fnmadd
  - ScalFunctions< Element >, [556](#)
  - Simd128\_impl< true, true, false, 2 >, [572](#)
  - Simd128\_impl< true, true, false, 4 >, [582](#)
  - Simd128\_impl< true, true, false, 8 >, [592](#)
  - Simd128\_impl< true, true, true, 2 >, [601](#)
  - Simd128\_impl< true, true, true, 4 >, [610](#)
  - Simd128\_impl< true, true, true, 8 >, [619](#)
  - Simd256\_impl< true, false, true, 8 >, [629](#)
  - Simd256\_impl< true, true, false, 2 >, [640](#)
  - Simd256\_impl< true, true, false, 4 >, [657](#)
  - Simd256\_impl< true, true, false, 8 >, [668](#)
  - Simd256\_impl< true, true, true, 2 >, [676](#)
  - Simd256\_impl< true, true, true, 4 >, [686](#), [693](#)
  - Simd256\_impl< true, true, true, 8 >, [702](#)
  - Simd512\_impl< true, false, true, 8 >, [711](#)
  - Simd512\_impl< true, true, false, 8 >, [721](#)
  - Simd512\_impl< true, true, true, 8 >, [730](#)
- fnmaddin
  - ScalFunctions< Element >, [557](#)
  - Simd128\_impl< true, true, false, 2 >, [573](#)
  - Simd128\_impl< true, true, false, 4 >, [582](#)
  - Simd128\_impl< true, true, false, 8 >, [592](#)
  - Simd128\_impl< true, true, true, 2 >, [601](#)
  - Simd128\_impl< true, true, true, 4 >, [610](#)
  - Simd128\_impl< true, true, true, 8 >, [619](#)
  - Simd256\_impl< true, false, true, 8 >, [629](#)
  - Simd256\_impl< true, true, false, 2 >, [640](#)
  - Simd256\_impl< true, true, false, 4 >, [657](#)
  - Simd256\_impl< true, true, false, 8 >, [668](#)
  - Simd256\_impl< true, true, true, 2 >, [676](#)
  - Simd256\_impl< true, true, true, 4 >, [687](#), [693](#)
  - Simd256\_impl< true, true, true, 8 >, [702](#)
  - Simd512\_impl< true, false, true, 8 >, [711](#)
  - Simd512\_impl< true, true, false, 8 >, [722](#)
  - Simd512\_impl< true, true, true, 8 >, [730](#)
- fnmaddx
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, [562](#)
  - Simd128\_impl< true, true, false, 2 >, [568](#)
  - Simd128\_impl< true, true, false, 4 >, [578](#)
  - Simd128\_impl< true, true, false, 8 >, [588](#)
  - Simd128\_impl< true, true, true, 2 >, [601](#)
  - Simd128\_impl< true, true, true, 4 >, [610](#)
  - Simd128\_impl< true, true, true, 8 >, [619](#)
- Simd256\_impl< true, true, false, 2 >, [636](#)
- Simd256\_impl< true, true, false, 4 >, [647](#), [650](#)
- Simd256\_impl< true, true, false, 8 >, [664](#)
- Simd256\_impl< true, true, true, 2 >, [676](#)
- Simd256\_impl< true, true, true, 4 >, [687](#), [693](#)
- Simd256\_impl< true, true, true, 8 >, [702](#)
- Simd512\_impl< true, true, false, 8 >, [717](#)
- Simd512\_impl< true, true, true, 8 >, [730](#)
- fnmaddxin
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, [562](#)
  - Simd128\_impl< true, true, false, 2 >, [569](#)
  - Simd128\_impl< true, true, false, 4 >, [579](#)
  - Simd128\_impl< true, true, false, 8 >, [589](#)
  - Simd128\_impl< true, true, true, 2 >, [601](#)
  - Simd128\_impl< true, true, true, 4 >, [610](#)
  - Simd128\_impl< true, true, true, 8 >, [619](#)
  - Simd256\_impl< true, true, false, 2 >, [636](#)
  - Simd256\_impl< true, true, false, 4 >, [647](#), [650](#)
  - Simd256\_impl< true, true, false, 8 >, [664](#)
  - Simd256\_impl< true, true, true, 2 >, [676](#)
  - Simd256\_impl< true, true, true, 4 >, [687](#), [693](#)
  - Simd256\_impl< true, true, true, 8 >, [702](#)
  - Simd512\_impl< true, true, false, 8 >, [717](#)
  - Simd512\_impl< true, true, true, 8 >, [730](#)
- FOR1D
  - parallel.h, [1035](#)
- FOR2D
  - parallel.h, [1035](#)
- FORBLOCK1D
  - parallel.h, [1034](#)
- FORBLOCK2D
  - parallel.h, [1035](#)
- ForceCheck\_charpoly
  - FFPACK, [304](#)
- ForceCheck\_Det
  - FFPACK, [304](#)
- ForceCheck\_fgemm
  - FFLAS, [72](#)
- ForceCheck\_ftdsm
  - FFLAS, [72](#)
- ForceCheck\_invert
  - FFPACK, [304](#)
- ForceCheck\_PLUQ
  - FFPACK, [303](#)
- ForStrategy1D
  - ForStrategy1D< blocksize\_t, Cut, Param >, [461](#)
- ForStrategy1D< blocksize\_t, Cut, Param >, [461](#)
  - begin, [462](#)
  - blockindex, [462](#)
  - build, [461](#)
  - changeBS, [463](#)
  - current, [462](#)
  - end, [462](#)
  - firstBlockSize, [462](#)
  - ForStrategy1D, [461](#)
  - ibeg, [462](#)
  - iend, [462](#)

- initialize, 461
- isTerminated, 461
- lastBlockSize, 462
- numBlock, 463
- numblocks, 462
- operator++, 462
- ForStrategy2D
  - ForStrategy2D< blocksize\_t, Cut, Param >, 464
- ForStrategy2D< blocksize\_t, Cut, Param >, 463
  - \_ibeg, 465
  - \_iend, 465
  - \_jbeg, 465
  - \_jend, 465
  - blockindex, 465
  - BLOCKS, 466
  - changeCBS, 466
  - changeRBS, 466
  - colblockindex, 465
  - colBlockSize, 466
  - colnumblocks, 465
  - current, 466
  - ForStrategy2D, 464
  - ibegin, 464
  - iend, 464
  - initialize, 464
  - isTerminated, 464
  - jbegin, 464
  - jend, 464
  - lastCBS, 466
  - lastRBS, 466
  - numColBlock, 466
  - numRowBlock, 466
  - operator<<, 465
  - operator++, 464
  - rowblockindex, 465
  - rowBlockSize, 466
  - rownumblocks, 464
- frand
  - FFLAS, 131, 135
- freduce
  - FFLAS, 107–111, 157, 167
  - FFLAS::details, 201, 202
- freduce\_1\_modular\_double
  - fflas\_c.h, 954
  - fflas\_lvl1.C, 974
- freduce\_2\_modular\_double
  - fflas\_c.h, 959
  - fflas\_lvl2.C, 979
- freduce\_constoverride
  - FFLAS, 108, 110
- freducein\_1\_modular\_double
  - fflas\_c.h, 954
  - fflas\_lvl1.C, 974
- freducein\_2\_modular\_double
  - fflas\_c.h, 958
  - fflas\_lvl2.C, 979
- freivalds
  - FFLAS, 111
- fscal
  - FFLAS, 112–116, 161, 169
  - FFLAS::details, 202, 203
- fscal\_1\_modular\_double
  - fflas\_c.h, 956
  - fflas\_lvl1.C, 975
- fscal\_2\_modular\_double
  - fflas\_c.h, 959
  - fflas\_lvl2.C, 980
- fscaln
  - FFLAS, 111, 113–116, 161, 169
  - FFLAS::details, 202, 203
- fscaln\_1\_modular\_double
  - fflas\_c.h, 956
  - fflas\_lvl1.C, 975
- fscaln\_2\_modular\_double
  - fflas\_c.h, 959
  - fflas\_lvl2.C, 980
- fspmm
  - FFLAS, 145
  - FFLAS::sparse\_details, 230–232
  - FFLAS::sparse\_details\_impl, 245, 246, 253, 254, 259, 263, 264, 272, 273
- fspmm\_dispatch
  - FFLAS::sparse\_details, 229
- fspmm\_mone
  - FFLAS::sparse\_details\_impl, 247, 254, 264, 265
- fspmm\_mone\_simd\_aligned
  - FFLAS::sparse\_details\_impl, 248, 255, 266
- fspmm\_mone\_simd\_unaligned
  - FFLAS::sparse\_details\_impl, 248, 255, 266
- fspmm\_one
  - FFLAS::sparse\_details\_impl, 247, 254, 264, 265
- fspmm\_one\_simd\_aligned
  - FFLAS::sparse\_details\_impl, 247, 255, 265
- fspmm\_one\_simd\_unaligned
  - FFLAS::sparse\_details\_impl, 247, 255, 265
- fspmm\_simd\_aligned
  - FFLAS::sparse\_details\_impl, 246, 253
- fspmm\_simd\_unaligned
  - FFLAS::sparse\_details\_impl, 246, 254
- fspmv
  - FFLAS, 145, 152
  - FFLAS::sparse\_details, 227–229, 236, 237
  - FFLAS::sparse\_details\_impl, 248, 249, 256, 260, 266, 267, 269, 270, 273–276
- fspmv\_dispatch
  - FFLAS::sparse\_details, 227
- fspmv\_mone
  - FFLAS::sparse\_details\_impl, 249, 256, 257, 267, 270, 271, 276, 277
- fspmv\_mone\_simd
  - FFLAS::sparse\_details\_impl, 271, 277
- fspmv\_one
  - FFLAS::sparse\_details\_impl, 249, 256, 257, 267, 270, 276, 277
- fspmv\_one\_simd
  - FFLAS::sparse\_details\_impl, 271, 277

- fspmv\_simd
  - FFLAS::sparse\_details\_impl, 269, 270, 275, 276
- fsquare
  - FFLAS, 91, 92, 178
- fsquare\_3\_modular\_double
  - fflas\_c.h, 963
  - fflas\_lvl3.C, 984
- fsquareCommon
  - FFLAS::Protected, 215
- fsub
  - FFLAS, 78, 80, 164, 171
- fsub\_1\_modular\_double
  - fflas\_c.h, 957
  - fflas\_lvl1.C, 976
- fsub\_2\_modular\_double
  - fflas\_c.h, 960
  - fflas\_lvl2.C, 981
- fsubin
  - FFLAS, 78, 81, 172
- fsubin\_1\_modular\_double
  - fflas\_c.h, 957
  - fflas\_lvl1.C, 977
- fsubin\_2\_modular\_double
  - fflas\_c.h, 960
  - fflas\_lvl2.C, 981
- fswap
  - FFLAS, 133, 163
- fswap\_1\_modular\_double
  - fflas\_c.h, 956
  - fflas\_lvl1.C, 976
- fsyr2k
  - FFLAS, 116
- fsyrk
  - FFLAS, 117–124
- fsyrk.C, 775
  - CUBE, 775
  - GFOPS, 775
  - main, 775
- fsyrk\_convert
  - FFLAS::Protected, 218
- fsyrk\_strassen
  - FFLAS, 124, 142
- fsytrf
  - FFPACK, 312, 313
- fsytrf.C, 775
  - CUBE, 776
  - GFOPS, 776
  - main, 776
- fsytrf\_BC\_Crout
  - FFPACK, 352
- fsytrf\_BC\_RL
  - FFPACK, 352
- fsytrf\_LOW\_RPM\_BC\_Crout
  - FFPACK, 353
- fsytrf\_nonunit
  - FFPACK, 313, 353, 354
- fsytrf\_RPM
  - FFPACK, 354
- fsytrf\_UP\_RPM
  - FFPACK, 353
- fsytrf\_UP\_RPM\_BC\_Crout
  - FFPACK, 353
- fsytrf\_UP\_RPM\_BC\_RL
  - FFPACK, 352
- ftmrm
  - FFLAS, 124, 125, 175
- ftmrm\_3\_modular\_double
  - fflas\_c.h, 962
  - fflas\_lvl3.C, 983
- ftmrmLeftLowerNoTransNonUnit< Element >, 467
- ftmrmLeftLowerNoTransUnit< Element >, 467
- ftmrmLeftLowerTransNonUnit< Element >, 467
- ftmrmLeftLowerTransUnit< Element >, 467
- ftmrmLeftUpperNoTransNonUnit< Element >, 467
- ftmrmLeftUpperNoTransUnit< Element >, 467
- ftmrmLeftUpperTransNonUnit< Element >, 467
- ftmrmLeftUpperTransUnit< Element >, 468
- ftmrmRightLowerNoTransNonUnit< Element >, 468
- ftmrmRightLowerNoTransUnit< Element >, 468
- ftmrmRightLowerTransNonUnit< Element >, 468
- ftmrmRightLowerTransUnit< Element >, 468
- ftmrmRightUpperNoTransNonUnit< Element >, 468
- ftmrmRightUpperNoTransUnit< Element >, 468
- ftmrmRightUpperTransNonUnit< Element >, 468
- ftmrmRightUpperTransUnit< Element >, 469
- ftmrv
  - FFLAS, 140
- ftsm
  - FFLAS, 126, 127, 141, 144, 145, 175
- ftsm\_3\_modular\_double
  - fflas\_c.h, 962
  - fflas\_lvl3.C, 983
- ftsmLeftLowerNoTransNonUnit< Element >, 469
- ftsmLeftLowerNoTransUnit< Element >, 469
- ftsmLeftLowerTransNonUnit< Element >, 469
- ftsmLeftLowerTransUnit< Element >, 469
- ftsmLeftUpperNoTransNonUnit< Element >, 469
- ftsmLeftUpperNoTransUnit< Element >, 470
- ftsmLeftUpperTransNonUnit< Element >, 470
- ftsmLeftUpperTransUnit< Element >, 470
- ftsmRightLowerNoTransNonUnit< Element >, 470
- ftsmRightLowerNoTransUnit< Element >, 470
- ftsmRightLowerTransNonUnit< Element >, 470
- ftsmRightLowerTransUnit< Element >, 470
- ftsmRightUpperNoTransNonUnit< Element >, 470
- ftsmRightUpperNoTransUnit< Element >, 471
- ftsmRightUpperTransNonUnit< Element >, 471
- ftsmRightUpperTransUnit< Element >, 471
- ftssyr2k
  - FFPACK, 311
- ftstr
  - FFPACK, 311
- ftsv
  - FFLAS, 128, 174
- ftsv\_2\_modular\_double
  - fflas\_c.h, 961



- fflas\_lvl2.C, [982](#)
- fttrtri
  - FFPACK, [309](#), [366](#)
- fttrtri.C, [776](#)
  - CUBE, [776](#)
  - GFOPS, [776](#)
  - main, [777](#)
- fttrtri\_modular\_double
  - ffpack.C, [991](#)
  - ffpack\_c.h, [1013](#)
- fttrtrm
  - FFPACK, [310](#), [367](#)
- fttrtrm\_modular\_double
  - ffpack.C, [992](#)
  - ffpack\_c.h, [1013](#)
- fzero
  - FFLAS, [130](#), [133](#), [135](#), [159](#), [165](#)
- fzero\_1\_modular\_double
  - fflas\_c.h, [955](#)
  - fflas\_lvl1.C, [974](#)
- fzero\_2\_modular\_double
  - fflas\_c.h, [958](#)
  - fflas\_lvl2.C, [978](#)
- gather
  - Simd128\_impl< true, true, false, 2 >, [567](#), [570](#)
  - Simd128\_impl< true, true, false, 4 >, [577](#), [579](#)
  - Simd128\_impl< true, true, false, 8 >, [587](#), [589](#)
  - Simd128\_impl< true, true, true, 2 >, [597](#)
  - Simd128\_impl< true, true, true, 4 >, [606](#)
  - Simd128\_impl< true, true, true, 8 >, [615](#)
  - Simd256\_impl< true, false, true, 8 >, [626](#)
  - Simd256\_impl< true, true, false, 2 >, [634](#), [637](#)
  - Simd256\_impl< true, true, false, 4 >, [646](#), [648](#), [651](#)
  - Simd256\_impl< true, true, false, 8 >, [662](#), [665](#)
  - Simd256\_impl< true, true, true, 2 >, [672](#)
  - Simd256\_impl< true, true, true, 4 >, [683](#), [689](#)
  - Simd256\_impl< true, true, true, 8 >, [698](#)
  - Simd512\_impl< true, false, true, 8 >, [708](#)
  - Simd512\_impl< true, true, false, 8 >, [715](#), [718](#)
  - Simd512\_impl< true, true, true, 8 >, [726](#)
- GaussJordan
  - FFPACK::Protected, [393](#)
- GCC\_VERSION
  - fflas-ffpack-config.h, [824](#)
- gebp
  - FFLAS::details, [206](#)
- genData
  - benchmark-fgemv.C, [792](#)
- GenericTag, [471](#)
- genInputs
  - ScalFunctions< Element >, [554](#)
- genInputsWithZero
  - ScalFunctions< Element >, [554](#)
- get
  - Simd128\_impl< true, true, false, 8 >, [590](#)
  - Simd128\_impl< true, true, true, 8 >, [615](#)
  - Simd256\_impl< true, true, false, 8 >, [665](#)
  - Simd256\_impl< true, true, true, 8 >, [698](#)
- get\_default\_random\_generator
  - ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >, [559](#)
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, [561](#)
- getArgumentValue
  - FFLAS, [185](#)
- getDataType
  - FFLAS, [151](#), [152](#)
- getEchelonForm
  - FFPACK, [337](#), [338](#)
- getEchelonForm< FFLAS\_FIELD< FFLAS\_ELT > >
  - FFPACK, [375](#)
- getEchelonForm\_modular\_double
  - ffpack.C, [1003](#)
  - ffpack\_c.h, [1022](#)
- getEchelonFormin\_modular\_double
  - ffpack.C, [1003](#)
  - ffpack\_c.h, [1023](#)
- getEchelonTransform
  - FFPACK, [338](#)
- getEchelonTransform< FFLAS\_FIELD< FFLAS\_ELT > >
  - FFPACK, [375](#)
- getEchelonTransform\_modular\_double
  - ffpack.C, [1003](#)
  - ffpack\_c.h, [1023](#)
- getListArgs
  - args-parser.h, [1042](#)
- getLTBruhatGen
  - FFPACK, [342](#), [343](#)
- getReducedEchelonForm
  - FFPACK, [339](#), [340](#)
- getReducedEchelonForm< FFLAS\_FIELD< FFLAS\_ELT > >
  - FFPACK, [376](#)
- getReducedEchelonForm\_modular\_double
  - ffpack.C, [1004](#)
  - ffpack\_c.h, [1023](#)
- getReducedEchelonFormin\_modular\_double
  - ffpack.C, [1004](#)
  - ffpack\_c.h, [1024](#)
- getReducedEchelonTransform
  - FFPACK, [341](#)
- getReducedEchelonTransform< FFLAS\_FIELD< FFLAS\_ELT > >
  - FFPACK, [376](#)
- getReducedEchelonTransform\_modular\_double
  - ffpack.C, [1004](#)
  - ffpack\_c.h, [1024](#)
- getSeed
  - FFLAS, [189](#)
- getStat
  - FFLAS, [154](#)
- getStatus
  - TestOneMethod< Simd >, [766](#)

- getTestName
  - TestOneMethod< Simd >, 766
- getTLBSize
  - FFLAS, 189
- getTriangular
  - FFPACK, 336
- getTriangular< FFLAS\_FIELD< FFLAS\_ELT > >
  - FFPACK, 374, 375
- getTriangular\_modular\_double
  - ffpack.C, 1002
  - ffpack\_c.h, 1022
- getTriangularin\_modular\_double
  - ffpack.C, 1003
  - ffpack\_c.h, 1022
- getTridiagonal
  - FFPACK, 354
- gf2ModularBalanced
  - regression-check.C, 1054
- GFOPS
  - charpoly.C, 774
  - fsyrk.C, 775
  - fsytrf.C, 776
  - fttri.C, 776
  - pluq.C, 777
  - winograd.C, 778
- Givaro, 398
- GRAIN
  - benchmark-fgemm-rns.C, 788
- Grain, 471
- greater
  - ScalFunctions< Element >, 557
  - Simd128\_impl< true, true, false, 2 >, 568
  - Simd128\_impl< true, true, false, 4 >, 578
  - Simd128\_impl< true, true, false, 8 >, 588
  - Simd128\_impl< true, true, true, 2 >, 602
  - Simd128\_impl< true, true, true, 4 >, 611
  - Simd128\_impl< true, true, true, 8 >, 620
  - Simd256\_impl< true, false, true, 8 >, 630
  - Simd256\_impl< true, true, false, 2 >, 635
  - Simd256\_impl< true, true, false, 4 >, 646, 649
  - Simd256\_impl< true, true, false, 8 >, 663
  - Simd256\_impl< true, true, true, 2 >, 677
  - Simd256\_impl< true, true, true, 4 >, 688, 694
  - Simd256\_impl< true, true, true, 8 >, 703
  - Simd512\_impl< true, false, true, 8 >, 711
  - Simd512\_impl< true, true, false, 8 >, 716
  - Simd512\_impl< true, true, true, 8 >, 731
- greater\_eq
  - ScalFunctions< Element >, 557
  - Simd128\_impl< true, true, false, 2 >, 568
  - Simd128\_impl< true, true, false, 4 >, 578
  - Simd128\_impl< true, true, false, 8 >, 588
  - Simd128\_impl< true, true, true, 2 >, 602
  - Simd128\_impl< true, true, true, 4 >, 611
  - Simd128\_impl< true, true, true, 8 >, 620
  - Simd256\_impl< true, false, true, 8 >, 630
  - Simd256\_impl< true, true, false, 2 >, 635
  - Simd256\_impl< true, true, false, 4 >, 647, 649
- Simd256\_impl< true, true, false, 8 >, 663
- Simd256\_impl< true, true, true, 2 >, 677
- Simd256\_impl< true, true, true, 4 >, 688, 694
- Simd256\_impl< true, true, true, 8 >, 703
- Simd512\_impl< true, false, true, 8 >, 712
- Simd512\_impl< true, true, false, 8 >, 716
- Simd512\_impl< true, true, true, 8 >, 731
- hadd
  - Simd256\_impl< true, false, true, 8 >, 630
  - Simd512\_impl< true, false, true, 8 >, 712
- hadd\_to\_scal
  - Simd128\_impl< true, true, false, 2 >, 569
  - Simd128\_impl< true, true, false, 4 >, 579
  - Simd128\_impl< true, true, false, 8 >, 589
  - Simd128\_impl< true, true, true, 2 >, 602
  - Simd128\_impl< true, true, true, 4 >, 611
  - Simd128\_impl< true, true, true, 8 >, 620
  - Simd256\_impl< true, false, true, 8 >, 631
  - Simd256\_impl< true, true, false, 2 >, 636
  - Simd256\_impl< true, true, false, 4 >, 648, 650
  - Simd256\_impl< true, true, false, 8 >, 664
  - Simd256\_impl< true, true, true, 2 >, 678
  - Simd256\_impl< true, true, true, 4 >, 688, 694
  - Simd256\_impl< true, true, true, 8 >, 703
  - Simd512\_impl< true, false, true, 8 >, 712
  - Simd512\_impl< true, true, false, 8 >, 718
  - Simd512\_impl< true, true, true, 8 >, 731
- half\_t
  - Simd256\_impl< true, true, false, 2 >, 634
  - Simd256\_impl< true, true, false, 4 >, 645
  - Simd256\_impl< true, true, false, 8 >, 662
  - Simd256\_impl< true, true, true, 2 >, 671
  - Simd256\_impl< true, true, true, 4 >, 682
  - Simd256\_impl< true, true, true, 8 >, 697
  - Simd512\_impl< true, true, false, 8 >, 715
  - Simd512\_impl< true, true, true, 8 >, 725
- has\_equal
  - FFLAS, 73
- has\_minus
  - FFLAS, 73
- has\_minus\_eq
  - FFLAS, 73
- has\_minus\_eq\_impl< C >, 471
  - value, 472
- has\_minus\_impl< C >, 472
  - value, 472
- has\_mul
  - FFLAS, 73
- has\_mul\_eq
  - FFLAS, 74
- has\_mul\_eq\_impl< C >, 472
  - value, 472
- has\_mul\_impl< C >, 472
  - value, 473
- has\_operation< T >, 473
  - value, 473
- has\_plus
  - FFLAS, 73



- has\_plus\_eq
  - FFLAS, [73](#)
- has\_plus\_eq\_impl< C >, [473](#)
  - value, [473](#)
- has\_plus\_impl< C >, [473](#)
  - value, [474](#)
- HAVE\_BLAS
  - config.h, [817](#)
- HAVE\_CBLAS
  - config.h, [817](#)
- HAVE\_CXX11
  - config.h, [817](#)
- HAVE\_DLFCN\_H
  - config.h, [818](#)
- HAVE\_FLOAT\_H
  - config.h, [818](#)
- HAVE\_INT128
  - config.h, [818](#)
- HAVE\_INTPTR\_T
  - config.h, [818](#)
- HAVE\_LAPACK
  - config.h, [818](#)
- HAVE\_LIMITS\_H
  - config.h, [818](#)
- HAVE\_LITTLE\_ENDIAN
  - config.h, [818](#)
- HAVE\_PTHREAD\_H
  - config.h, [818](#)
- HAVE\_STDDEF\_H
  - config.h, [818](#)
- HAVE\_STDINT\_H
  - config.h, [818](#)
- HAVE\_STDIO\_H
  - config.h, [818](#)
- HAVE\_STDLIB\_H
  - config.h, [818](#)
- HAVE\_STRING\_H
  - config.h, [818](#)
- HAVE\_STRINGS\_H
  - config.h, [818](#)
- HAVE\_SYS\_STAT\_H
  - config.h, [818](#)
- HAVE\_SYS\_TIME\_H
  - config.h, [819](#)
- HAVE\_SYS\_TYPES\_H
  - config.h, [819](#)
- HAVE\_UNISTD\_H
  - config.h, [819](#)
- HelperFlag, [474](#)
  - aut, [474](#)
  - coo, [474](#)
  - csr, [474](#)
  - ell, [474](#)
  - none, [474](#)
  - pm1, [474](#)
- HelperMod
  - HelperMod< Field, ElementCategories::MachineIntTag >, [475](#)
  - HelperMod< Field, FFLAS::ElementCategories::ArbitraryPrecIntTag >, [476](#)
  - HelperMod< Field, FFLAS::ElementCategories::FixedPrecIntTag >, [476](#)
  - HelperMod< Field, FFLAS::ElementCategories::MachineFloatTag >, [477](#)
  - HelperMod< Field, ElementCategories::MachineIntTag >, [475](#)
  - HelperMod, [475](#)
  - invp, [475](#)
  - max, [475](#)
  - min, [475](#)
  - p, [475](#)
  - pow50rem, [475](#)
  - HelperMod< Field, ElementTraits >, [474](#)
  - HelperMod< Field, FFLAS::ElementCategories::ArbitraryPrecIntTag >, [476](#)
  - HelperMod, [476](#)
  - p, [476](#)
  - HelperMod< Field, FFLAS::ElementCategories::FixedPrecIntTag >, [476](#)
  - HelperMod, [476](#)
  - p, [477](#)
  - HelperMod< Field, FFLAS::ElementCategories::MachineFloatTag >, [477](#)
  - HelperMod, [477](#)
  - invp, [477](#)
  - max, [477](#)
  - min, [477](#)
  - p, [477](#)
- helpString
  - Argument, [403](#)
- HYB\_ZO
  - FFLAS, [76](#)
- hyb\_zo.h, [905](#)
- hyb\_zo\_pspmm.inl, [905](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_pspmm\_INL, [906](#)
- hyb\_zo\_pspmv.inl, [906](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_pspmv\_INL, [906](#)
- hyb\_zo\_spm.inl, [906](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_spm\_INL, [907](#)
- hyb\_zo\_spmv.inl, [907](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_spmv\_INL, [907](#)
- hyb\_zo\_utils.inl, [907](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_utils\_INL, [908](#)
- Hybrid, [478](#)
- Hybrid\_KGF\_LUK\_MinPoly
  - FFPACK::Protected, [396](#)
- ibeg
  - ForStrategy1D< blocksize\_t, Cut, Param >, [462](#)
- ibegin
  - ForStrategy2D< blocksize\_t, Cut, Param >, [464](#)
- idamax\_

- config-blas.h, [813](#)
- iend
  - ForStrategy1D< blocksize\_t, Cut, Param >, [462](#)
  - ForStrategy2D< blocksize\_t, Cut, Param >, [464](#)
- igebb11
  - FFLAS::details, [205](#)
- igebb14
  - FFLAS::details, [204](#)
- igebb21
  - FFLAS::details, [204](#)
- igebb24
  - FFLAS::details, [203](#)
- igebb41
  - FFLAS::details, [204](#)
- igebb44
  - FFLAS::details, [203](#)
- igebp
  - FFLAS::details, [205](#)
- igemm
  - FFLAS::Protected, [222](#)
- igemm.doxy, [859](#)
- igemm.h, [859](#)
- igemm.inl, [859](#)
  - \_\_FFLASFFPACK\_fflas\_igemm\_igemm\_INL, [860](#)
- igemm\_
  - FFLAS, [128](#)
- igemm\_colmajor
  - FFLAS::Protected, [221](#), [222](#)
- igemm\_kernels.h, [860](#)
- igemm\_kernels.inl, [861](#)
  - \_\_FFLASFFPACK\_fflas\_igemm\_igemm\_kernels\_INL, [861](#)
- igemm\_tools.h, [861](#)
- igemm\_tools.inl, [862](#)
  - \_\_FFLASFFPACK\_fflas\_igemm\_igemm\_tools\_INL, [862](#)
- in\_range
  - flimits.h, [1048](#), [1049](#)
- index\_t
  - fflas\_sparse.h, [885](#)
  - parallel.h, [1033](#)
- InfNorm
  - FFLAS, [77](#)
- Info, [478](#), [479](#)
  - begin, [479](#), [480](#)
  - Info, [478](#), [479](#)
  - operator=, [478](#), [480](#)
  - perm, [479](#), [480](#)
  - size, [479](#), [480](#)
- info
  - BlockTransposeSIMD< Field, Simd, >, [409](#)
- init
  - FieldSimd< \_Field >, [448](#)
  - rns\_double, [528–530](#)
  - rns\_double\_extended, [540](#)
  - RNSInteger< RNS >, [544](#)
  - RNSIntegerMod< RNS >, [548](#), [549](#)
- init\_transpose
  - rns\_double, [529](#)
- init\_y
  - FFLAS::sparse\_details, [226](#)
- initA
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, [500](#)
- initB
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, [500](#)
- initC
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, [500](#)
- initialize
  - ForStrategy1D< blocksize\_t, Cut, Param >, [461](#)
  - ForStrategy2D< blocksize\_t, Cut, Param >, [464](#)
- initOut
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, [500](#)
- INLINE
  - fflas\_simd.h, [872](#)
- inplace
  - Bench< Elt >, [409](#)
- inputs
  - TestOneMethod< Simd >, [767](#)
- INST\_OR\_DECL
  - fflas\_L1\_inst.C, [963](#)
  - fflas\_L1\_inst.h, [964](#)
  - fflas\_L2\_inst.C, [967](#)
  - fflas\_L2\_inst.h, [968](#)
  - fflas\_L3\_inst.C, [970](#)
  - fflas\_L3\_inst.h, [971](#)
  - ffpack\_inst.C, [1025](#)
  - ffpack\_inst.h, [1026](#)
- integer
  - rns\_double, [527](#)
  - rns\_double\_extended, [539](#)
  - RNSInteger< RNS >, [543](#)
  - RNSIntegerMod< RNS >, [547](#)
- Interfaces, [42](#)
- interfaces.doxy, [951](#)
- IntType
  - ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >::FloatingPointTestDistribution, [460](#)
- inv
  - RNSIntegerMod< RNS >, [550](#)
- Invert
  - FFPACK, [321](#), [369](#)
- Invert2
  - FFPACK, [322](#), [369](#)
- Invert2\_modular\_double
  - ffpack.C, [998](#)
  - ffpack\_c.h, [1018](#)
- Invert\_modular\_double
  - ffpack.C, [998](#)
  - ffpack\_c.h, [1018](#)
- Invertin\_modular\_double
  - ffpack.C, [998](#)

- ffpack\_c.h, 1018
- invp
  - HelperMod< Field, ElementCategories::MachineIntTags >, 475
  - HelperMod< Field, FFLAS::ElementCategories::MachineIntTags >, 477
- is\_all\_same< Args >, 480
- is\_all\_same< T, Args... >, 480
  - value, 480
- is\_all\_same<>, 480
  - value, 481
- is\_same\_element
  - Bench< Elt >, 407
  - NoSimd< T >, 521
  - Simd128\_impl< true, true, false, 2 >, 566
  - Simd128\_impl< true, true, false, 4 >, 576
  - Simd128\_impl< true, true, false, 8 >, 586
  - Simd128\_impl< true, true, true, 2 >, 597
  - Simd128\_impl< true, true, true, 4 >, 606
  - Simd128\_impl< true, true, true, 8 >, 615
  - Simd256\_impl< true, false, true, 8 >, 625
  - Simd256\_impl< true, true, false, 2 >, 633
  - Simd256\_impl< true, true, false, 4 >, 645, 650
  - Simd256\_impl< true, true, false, 8 >, 662
  - Simd256\_impl< true, true, true, 2 >, 672
  - Simd256\_impl< true, true, true, 4 >, 682, 695
  - Simd256\_impl< true, true, true, 8 >, 698
  - Simd512\_impl< true, false, true, 8 >, 707
  - Simd512\_impl< true, true, false, 8 >, 715
  - Simd512\_impl< true, true, true, 8 >, 726
  - Test< Elt >, 762
- is\_simd< T >, 481
  - type, 481
  - value, 481
- isMOne
  - RNSInteger< RNS >, 544
  - RNSIntegerMod< RNS >, 548
- isOdd
  - FFPACK, 377
- isOne
  - RNSInteger< RNS >, 543
  - RNSIntegerMod< RNS >, 548
- IsSingular
  - FFPACK, 326, 371
- IsSingular\_modular\_double
  - ffpack.C, 999
  - ffpack\_c.h, 1019
- isSparseMatrix< Field, M >, 481
- isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::COO >, 481
- isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::COO\_ZO >, 482
- isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::CSR >, 482
- isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::CSR\_HYB >, 482
- isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::CSR\_ZO >, 483
- isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL >, 483
- isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL\_simd >, 483
- isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL\_simd\_ZO >, 484
- isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL\_ZO >, 484
- isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::HYB\_ZO >, 484
- isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::SELL >, 484
- isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::SELL\_ZO >, 485
- isSparseMatrixMKLFormat< F, M >, 485
- isSparseMatrixSimdFormat< F, M >, 485
- isTerminated
  - ForStrategy1D< blocksize\_t, Cut, Param >, 461
  - ForStrategy2D< blocksize\_t, Cut, Param >, 464
- isZero
  - RNSInteger< RNS >, 544
  - RNSIntegerMod< RNS >, 548
- isZOSparseMatrix< F, M >, 486
- isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::COO\_ZO >, 486
- isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::CSR\_ZO >, 486
- isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL\_simd\_ZO >, 486
- isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL\_ZO >, 487
- isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::SELL\_ZO >, 487
- Iterative, 487
- iters
  - Bench< Elt >, 409
- jbegin
  - ForStrategy2D< blocksize\_t, Cut, Param >, 464
- jend
  - ForStrategy2D< blocksize\_t, Cut, Param >, 464
- kaapi\_routines.inl, 1032
  - \_\_FFLASFFPACK\_KAAPI\_ROUTINES\_INL, 1032
- KellerGehrig
  - FFPACK::Protected, 393
- KGFast
  - FFPACK::Protected, 394
- KGFast\_generalized
  - FFPACK::Protected, 394
- kmax
  - Sparse< \_Field, SparseMatrix\_t::COO >, 736
  - Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, 738
  - Sparse< \_Field, SparseMatrix\_t::CSR >, 739
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, 741
  - Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, 743
  - Sparse< \_Field, SparseMatrix\_t::ELL >, 744
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, 746

- Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, 748
- Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, 749
- Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, 751
- Sparse< \_Field, SparseMatrix\_t::SELL >, 753
- Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, 755
- KrylovElim
  - FFPACK, 371
- KrylovElim\_modular\_double
  - ffpack.C, 998
  - ffpack\_c.h, 1018
- lapack.C, 1053
  - \_\_FFLASFFPACK\_CONFIGURATION, 1053
  - \_\_FFLASFFPACK\_HAVE\_LAPACK, 1053
  - main, 1053
- LAPACKPerm2MathPerm
  - FFPACK, 304
  - ffpack.C, 988
  - ffpack\_c.h, 1009
- lastBlockSize
  - ForStrategy1D< blocksize\_t, Cut, Param >, 462
- lastCBS
  - ForStrategy2D< blocksize\_t, Cut, Param >, 466
- lastRBS
  - ForStrategy2D< blocksize\_t, Cut, Param >, 466
- launch\_fger
  - test-fger.C, 1069
- launch\_fger\_dispatch
  - test-fger.C, 1069
- launch\_MM
  - test-fgemm.C, 1065
- launch\_MM\_dispatch
  - test-fgemm-check.C, 1063
  - test-fgemm.C, 1065
- launch\_MV
  - test-fgemv.C, 1067
- launch\_MV\_dispatch
  - test-fgemv.C, 1067
- launch\_test
  - test-charpoly.C, 1055
  - test-lu.C, 1091
  - test-quasisep.C, 1097
- launch\_wino
  - benchmark-wino.C, 803
- LazyTag, 487
- LD
  - fflas\_transpose.h, 914
- ld
  - Sparse< \_Field, SparseMatrix\_t::ELL >, 745
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, 746
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, 748
  - Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, 750
- LeadingSubmatrixRankProfiles
  - FFPACK, 332
  - ffpack.C, 1001
  - ffpack\_c.h, 1021
- lesser
  - ScalFunctions< Element >, 557
  - Simd128\_impl< true, true, false, 2 >, 568
  - Simd128\_impl< true, true, false, 4 >, 578
  - Simd128\_impl< true, true, false, 8 >, 588
  - Simd128\_impl< true, true, true, 2 >, 602
  - Simd128\_impl< true, true, true, 4 >, 611
  - Simd128\_impl< true, true, true, 8 >, 620
  - Simd256\_impl< true, false, true, 8 >, 629
  - Simd256\_impl< true, true, false, 2 >, 635
  - Simd256\_impl< true, true, false, 4 >, 646, 649
  - Simd256\_impl< true, true, false, 8 >, 663
  - Simd256\_impl< true, true, true, 2 >, 677
  - Simd256\_impl< true, true, true, 4 >, 688, 694
  - Simd256\_impl< true, true, true, 8 >, 703
  - Simd512\_impl< true, false, true, 8 >, 711
  - Simd512\_impl< true, true, false, 8 >, 716
  - Simd512\_impl< true, true, true, 8 >, 731
- lesser\_eq
  - ScalFunctions< Element >, 557
  - Simd128\_impl< true, true, false, 2 >, 568
  - Simd128\_impl< true, true, false, 4 >, 578
  - Simd128\_impl< true, true, false, 8 >, 588
  - Simd128\_impl< true, true, true, 2 >, 602
  - Simd128\_impl< true, true, true, 4 >, 611
  - Simd128\_impl< true, true, true, 8 >, 620
  - Simd256\_impl< true, false, true, 8 >, 629
  - Simd256\_impl< true, true, false, 2 >, 635
  - Simd256\_impl< true, true, false, 4 >, 647, 649
  - Simd256\_impl< true, true, false, 8 >, 663
  - Simd256\_impl< true, true, true, 2 >, 677
  - Simd256\_impl< true, true, true, 4 >, 688, 694
  - Simd256\_impl< true, true, true, 8 >, 703
  - Simd512\_impl< true, false, true, 8 >, 711
  - Simd512\_impl< true, true, false, 8 >, 717
  - Simd512\_impl< true, true, true, 8 >, 731
- limits< char >, 488
  - digits, 488
  - max, 488
  - min, 488
  - T, 488
- limits< double >, 488
  - digits, 489
  - max, 489
  - min, 489
  - T, 489
- limits< float >, 489
  - digits, 489
  - max, 489
  - min, 489
  - T, 489
- limits< Givaro::Integer >, 490
  - max, 490
  - min, 490
  - T, 490
- limits< int >, 490
  - digits, 491
  - max, 491
  - min, 491

- T, [490](#)
- limits< long >, [491](#)
  - digits, [491](#)
  - max, [491](#)
  - min, [491](#)
  - T, [491](#)
- limits< long long >, [491](#)
  - digits, [492](#)
  - max, [492](#)
  - min, [492](#)
  - T, [492](#)
- limits< Reclnt::rint< K > >, [492](#)
  - max, [492](#)
  - min, [492](#)
  - T, [492](#)
- limits< Reclnt::ruint< K > >, [493](#)
  - max, [493](#)
  - min, [493](#)
  - T, [493](#)
- limits< short int >, [493](#)
  - digits, [494](#)
  - max, [494](#)
  - min, [494](#)
  - T, [494](#)
- limits< signed char >, [494](#)
  - digits, [494](#)
  - max, [494](#)
  - min, [494](#)
  - T, [494](#)
- limits< T >, [488](#)
- limits< unsigned char >, [495](#)
  - digits, [495](#)
  - max, [495](#)
  - min, [495](#)
  - T, [495](#)
- limits< unsigned int >, [495](#)
  - digits, [496](#)
  - max, [496](#)
  - min, [496](#)
  - T, [495](#)
- limits< unsigned long >, [496](#)
  - digits, [496](#)
  - max, [496](#)
  - min, [496](#)
  - T, [496](#)
- limits< unsigned long long >, [496](#)
  - digits, [497](#)
  - max, [497](#)
  - min, [497](#)
  - T, [497](#)
- limits< unsigned short int >, [497](#)
  - digits, [497](#)
  - max, [497](#)
  - min, [497](#)
  - T, [497](#)
- load
  - Simd128\_impl< true, true, false, 2 >, [567](#), [570](#)
  - Simd128\_impl< true, true, false, 4 >, [577](#), [580](#)
  - Simd128\_impl< true, true, false, 8 >, [587](#), [590](#)
  - Simd128\_impl< true, true, true, 2 >, [597](#)
  - Simd128\_impl< true, true, true, 4 >, [606](#)
  - Simd128\_impl< true, true, true, 8 >, [615](#)
  - Simd256\_impl< true, false, true, 8 >, [626](#)
  - Simd256\_impl< true, true, false, 2 >, [634](#), [637](#)
  - Simd256\_impl< true, true, false, 4 >, [646](#), [648](#), [652](#)
  - Simd256\_impl< true, true, false, 8 >, [662](#), [665](#)
  - Simd256\_impl< true, true, true, 2 >, [672](#)
  - Simd256\_impl< true, true, true, 4 >, [683](#), [689](#)
  - Simd256\_impl< true, true, true, 8 >, [699](#)
  - Simd512\_impl< true, false, true, 8 >, [708](#)
  - Simd512\_impl< true, true, false, 8 >, [716](#), [719](#)
  - Simd512\_impl< true, true, true, 8 >, [727](#)
- loadu
  - Simd128\_impl< true, true, false, 2 >, [567](#), [570](#)
  - Simd128\_impl< true, true, false, 4 >, [577](#), [580](#)
  - Simd128\_impl< true, true, false, 8 >, [587](#), [590](#)
  - Simd128\_impl< true, true, true, 2 >, [597](#)
  - Simd128\_impl< true, true, true, 4 >, [606](#)
  - Simd128\_impl< true, true, true, 8 >, [616](#)
  - Simd256\_impl< true, false, true, 8 >, [626](#)
  - Simd256\_impl< true, true, false, 2 >, [634](#), [637](#)
  - Simd256\_impl< true, true, false, 4 >, [646](#), [648](#), [652](#)
  - Simd256\_impl< true, true, false, 8 >, [663](#), [665](#)
  - Simd256\_impl< true, true, true, 2 >, [673](#)
  - Simd256\_impl< true, true, true, 4 >, [683](#), [689](#)
  - Simd256\_impl< true, true, true, 8 >, [699](#)
  - Simd512\_impl< true, false, true, 8 >, [708](#)
  - Simd512\_impl< true, true, false, 8 >, [716](#), [719](#)
  - Simd512\_impl< true, true, true, 8 >, [727](#)
- LQUPtoInverseOfFullRankMinor
  - FFPACK, [346](#), [377](#)
- LT\_OBJDIR
  - config.h, [819](#)
- LTBruhatGen
  - FFPACK, [342](#)
- LTQSorter
  - FFPACK, [343](#)
- LUdivine
  - FFPACK, [315](#), [355](#), [367](#)
- LUdivine\_construct
  - FFPACK::Protected, [392](#), [398](#)
- LUdivine\_gauss
  - FFPACK, [354](#), [368](#)
- LUdivine\_gauss\_modular\_double
  - ffpack\_c.h, [1014](#)
- LUdivine\_modular\_double
  - ffpack.C, [992](#)
  - ffpack\_c.h, [1013](#)
- LUdivine\_small
  - FFPACK, [355](#), [367](#)
- LUdivine\_small\_modular\_double
  - ffpack\_c.h, [1014](#)
- LUKrylov
  - FFPACK::Protected, [394](#)

LUKrylov\_KGFast

FFPACK::Protected, 395

m

Bench< Elt >, 408

Sparse< \_Field, SparseMatrix\_t::COO >, 736

Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, 738

Sparse< \_Field, SparseMatrix\_t::CSR >, 739

Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, 741

Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, 743

Sparse< \_Field, SparseMatrix\_t::ELL >, 745

Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, 746

Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, 748

Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, 749

Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, 751

Sparse< \_Field, SparseMatrix\_t::SELL >, 753

Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, 755

MachineFloatTag, 498

MachineIntTag, 498

main

101-fgemm.C, 1105

2x2-fgemm.C, 1105

2x2-ftrsv.C, 1105

2x2-pluq.C, 1106

arithprog.C, 773

benchmark-charpoly-mp.C, 779

benchmark-charpoly.C, 780

benchmark-checkers.C, 781

benchmark-dgemm.C, 782

benchmark-dgetrf.C, 783

benchmark-dgetri.C, 783

benchmark-dsytrf.C, 784

benchmark-dtrsm.C, 784

benchmark-dtrtri.C, 785

benchmark-fadd-lvl2.C, 785

benchmark-fdot.C, 786

benchmark-fgemm-mp.C, 787

benchmark-fgemm-rns.C, 789

benchmark-fgemm.C, 789

benchmark-fgemv-mp.C, 790

benchmark-fgemv.C, 794

benchmark-fgesv.C, 794

benchmark-fsyr2k.C, 795

benchmark-fsyrk.C, 795

benchmark-fsytrf.C, 796

benchmark-ftrsm-mp.C, 797

benchmark-ftrsm.C, 797

benchmark-ftrsv.C, 798

benchmark-ftrtri.C, 798

benchmark-inverse.C, 799

benchmark-lqup-mp.C, 799

benchmark-lqup.C, 800

benchmark-pluq.C, 801

benchmark-quasisep.C, 802

benchmark-storage-transpose.C, 803

benchmark-wino.C, 803

cblas.C, 1051

charpoly.C, 774

clapack.C, 1051

cuda.C, 1052

det.C, 804

fblas.C, 1052

fflas-101\_1.C, 1106

fflas-101\_3.C, 1107

fflas\_101.C, 1107

fflas\_101\_lvl1.C, 1107

ffpack-fgesv.C, 1108

ffpack-solve.C, 1108

fsyrk.C, 775

fsytrf.C, 776

ftrtri.C, 777

lapack.C, 1053

matmul.C, 804

pluq.C, 777, 778

rank.C, 805

regression-check.C, 1054

solve.C, 805

test-charpoly-check.C, 1055

test-charpoly.C, 1056

test-compressQ.C, 1056

test-det-check.C, 1057

test-det.C, 1058

test-echelon.C, 1060

test-fadd.C, 1061

test-fdot.C, 1062

test-fgemm-check.C, 1064

test-fgemm.C, 1066

test-fgemv.C, 1068

test-fger.C, 1069

test-fgesv.C, 1071

test-finit.C, 1072

test-fscal.C, 1073

test-fsyr2k.C, 1074

test-fsyrk.C, 1076

test-fsytrf.C, 1078

test-ftrmm.C, 1079

test-ftrmv.C, 1080

test-ftrsm-check.C, 1080

test-ftrsm.C, 1082

test-ftrssyr2k.C, 1083

test-ftrstr.C, 1084

test-ftrsv.C, 1085

test-ftrtri.C, 1086

test-interfaces-c.c, 1087

test-invert-check.C, 1087

test-io.C, 1088

test-lu.C, 1091

test-maxdelayeddim.C, 1092

test-minpoly.C, 1093

test-multifile2.C, 1094

test-nullspace.C, 1095

test-permutations.C, 1096

test-pluq-check.C, 1097

test-quasisep.C, 1098

test-rankprofiles.C, 1099



- test-rpm.C, [1100](#)
- test-simd.C, [1103](#)
- test-solve.C, [1104](#)
- test-storage-transpose.C, [1104](#)
- winograd.C, [779](#)
- mainpage.doxy, [804](#)
- mask\_high
  - Simd128\_impl< true, true, false, 8 >, [593](#)
  - Simd128\_impl< true, true, true, 8 >, [620](#)
  - Simd256\_impl< true, true, false, 8 >, [668](#)
  - Simd256\_impl< true, true, true, 8 >, [703](#)
  - Simd512\_impl< true, true, false, 8 >, [722](#)
  - Simd512\_impl< true, true, true, 8 >, [732](#)
- mask\_t
  - read\_sparse.h, [909](#)
- maskstore
  - Simd512\_impl< true, true, false, 8 >, [716](#), [719](#)
  - Simd512\_impl< true, true, true, 8 >, [727](#)
- MatF2MatD\_Triangular
  - FFLAS::Protected, [222](#)
- MatF2MatFI\_Triangular
  - FFLAS::Protected, [223](#)
- MathPerm2LAPACKPerm
  - FFPACK, [304](#)
  - ffpack.C, [988](#)
  - ffpack\_c.h, [1009](#)
- Matio.h, [1049](#)
  - read\_field, [1049](#)
  - write\_field, [1049](#)
- matmul.C, [804](#)
  - main, [804](#)
- matmul.doxy, [839](#)
- Matrix Multiplication Algorithms, [42](#)
- MatrixApplyS
  - FFPACK, [357](#), [358](#)
- MatrixApplyS\_modular\_double
  - ffpack.C, [988](#)
  - ffpack\_c.h, [1010](#)
- MatrixApplyT
  - FFPACK, [359](#)
- MatrixApplyT\_modular\_double
  - ffpack.C, [988](#)
  - ffpack\_c.h, [1010](#)
- MatVecMinPoly
  - FFPACK, [325](#), [370](#)
  - FFPACK::Protected, [396](#)
- max
  - HelperMod< Field, ElementCategories::MachineIntTag >, [475](#)
  - HelperMod< Field, FFLAS::ElementCategories::MachineIntTag >, [477](#)
  - limits< char >, [488](#)
  - limits< double >, [489](#)
  - limits< float >, [489](#)
  - limits< Givaro::Integer >, [490](#)
  - limits< int >, [491](#)
  - limits< long >, [491](#)
  - limits< long long >, [492](#)
  - limits< RecInt::rint< K > >, [492](#)
  - limits< RecInt::ruint< K > >, [493](#)
  - limits< short int >, [494](#)
  - limits< signed char >, [494](#)
  - limits< unsigned char >, [495](#)
  - limits< unsigned int >, [496](#)
  - limits< unsigned long >, [496](#)
  - limits< unsigned long long >, [497](#)
  - limits< unsigned short int >, [497](#)
- max3
  - FFLAS, [77](#)
- max4
  - FFLAS, [77](#)
- MAX\_THREADS
  - parallel.h, [1034](#)
- MAX\_WITH\_SIZE\_T
  - test-maxdelayeddim.C, [1092](#)
- maxCardinality
  - FFLAS, [154](#)
- maxCardinality< Givaro::Modular< int32\_t > >
  - FFLAS, [154](#)
- maxCardinality< Givaro::Modular< int64\_t > >
  - FFLAS, [154](#)
- maxCol
  - StatsMatrix, [758](#)
- maxColDifference
  - StatsMatrix, [758](#)
- MaxDelayedDim
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [500](#)
- maxElement
  - RNSIntegerMod< RNS >, [548](#)
- maxpy
  - FieldSimd< \_Field >, [452](#)
- maxpyin
  - FieldSimd< \_Field >, [452](#)
- maxRow
  - StatsMatrix, [757](#)
- maxrow
  - Sparse< \_Field, SparseMatrix\_t::COO >, [737](#)
  - Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, [738](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR >, [740](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, [741](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, [743](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL >, [745](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, [746](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, [748](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, [750](#)
  - Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, [751](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL >, [753](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, [755](#)
- maxRowDifference
  - StatsMatrix, [758](#)
- MaxStorableValue
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [503](#)





- Bmin, [502](#)
- Bunfit, [501](#)
- checkA, [501](#)
- checkB, [501](#)
- checkOut, [501](#)
- Cmax, [502](#)
- Cmin, [502](#)
- DelayedField, [499](#)
- delayedField, [503](#)
- DelayedField\_t, [499](#)
- DFElt, [499](#)
- FieldMax, [502](#)
- FieldMin, [502](#)
- initA, [500](#)
- initB, [500](#)
- initC, [500](#)
- initOut, [500](#)
- MaxDelayedDim, [500](#)
- MaxStorableValue, [503](#)
- MMHelper, [499](#), [500](#)
- operator<<, [502](#)
- Outmax, [503](#)
- Outmin, [503](#)
- parseq, [503](#)
- recLevel, [502](#)
- Self\_t, [499](#)
- setOutBounds, [501](#)
- mod
  - FieldSimd< \_Field >, [450](#)
  - Simd128\_impl< true, true, false, 2 >, [573](#)
  - Simd128\_impl< true, true, false, 4 >, [583](#)
  - Simd128\_impl< true, true, false, 8 >, [593](#)
  - Simd128\_impl< true, true, true, 2 >, [602](#)
  - Simd128\_impl< true, true, true, 4 >, [611](#)
  - Simd128\_impl< true, true, true, 8 >, [620](#)
  - Simd256\_impl< true, false, true, 8 >, [631](#)
  - Simd256\_impl< true, true, false, 2 >, [641](#)
  - Simd256\_impl< true, true, false, 4 >, [658](#)
  - Simd256\_impl< true, true, false, 8 >, [669](#)
  - Simd256\_impl< true, true, true, 2 >, [678](#)
  - Simd256\_impl< true, true, true, 4 >, [688](#), [694](#)
  - Simd256\_impl< true, true, true, 8 >, [704](#)
  - Simd512\_impl< true, true, false, 8 >, [722](#)
  - Simd512\_impl< true, true, true, 8 >, [732](#)
- MODE
  - parallel.h, [1036](#)
- ModeTraits< Field >, [512](#)
  - value, [512](#)
- ModeTraits< Givaro::Modular< Element, Compute > >, [513](#)
  - value, [513](#)
- ModeTraits< Givaro::Modular< Givaro::Integer, Compute > >, [513](#)
  - value, [513](#)
- ModeTraits< Givaro::Modular< int16\_t, Compute > >, [513](#)
  - value, [513](#)
- ModeTraits< Givaro::Modular< int32\_t, Compute > >, [514](#)
  - value, [514](#)
- ModeTraits< Givaro::Modular< int64\_t, uint64\_t > >, [514](#)
  - value, [514](#)
- ModeTraits< Givaro::Modular< int8\_t, Compute > >, [514](#)
  - value, [514](#)
- ModeTraits< Givaro::Modular< ReclInt::ruint< K >, Compute > >, [515](#)
  - value, [515](#)
- ModeTraits< Givaro::Modular< uint16\_t, Compute > >, [515](#)
  - value, [515](#)
- ModeTraits< Givaro::Modular< uint32\_t, Compute > >, [515](#)
  - value, [515](#)
- ModeTraits< Givaro::Modular< uint8\_t, Compute > >, [516](#)
  - value, [516](#)
- ModeTraits< Givaro::ModularBalanced< Element > >, [516](#)
  - value, [516](#)
- ModeTraits< Givaro::ModularBalanced< Givaro::Integer > >, [516](#)
  - value, [516](#)
- ModeTraits< Givaro::ModularBalanced< int16\_t > >, [517](#)
  - value, [517](#)
- ModeTraits< Givaro::ModularBalanced< int32\_t > >, [517](#)
  - value, [517](#)
- ModeTraits< Givaro::ModularBalanced< int8\_t > >, [517](#)
  - value, [517](#)
- ModeTraits< Givaro::Montgomery< T > >, [517](#)
  - value, [518](#)
- ModeTraits< Givaro::ZRing< double > >, [518](#)
  - value, [518](#)
- ModeTraits< Givaro::ZRing< float > >, [518](#)
  - value, [518](#)
- ModeTraits< Givaro::ZRing< Givaro::Integer > >, [518](#)
  - value, [519](#)
- ModField
  - rns\_double, [527](#)
  - rns\_double\_extended, [539](#)
  - RNSIntegerMod< RNS >, [547](#)
- modp
  - FFLAS::vectorised, [283](#)
  - FFLAS::vectorised::unswitch, [285](#)
- ModularBalanced< T >, [519](#)
- ModularTag, [519](#)
- mOne
  - RNSInteger< RNS >, [545](#)
  - RNSIntegerMod< RNS >, [552](#)
- mone
  - FFLAS, [76](#)

- Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, [751](#)
- MonotonicApplyP
  - FFPACK, [306](#)
- MonotonicCompress
  - FFPACK, [356](#)
- MonotonicCompressCycles
  - FFPACK, [356](#)
- MonotonicCompressMorePivots
  - FFPACK, [356](#)
- MonotonicExpand
  - FFPACK, [357](#)
- Montgomery< T >, [519](#)
- mul
  - FieldSimd< \_Field >, [450](#)
  - RNSIntegerMod< RNS >, [550](#)
  - ScalFunctions< Element >, [556](#)
  - Simd128\_impl< true, true, false, 2 >, [572](#)
  - Simd128\_impl< true, true, false, 4 >, [582](#)
  - Simd128\_impl< true, true, false, 8 >, [592](#)
  - Simd128\_impl< true, true, true, 2 >, [600](#)
  - Simd128\_impl< true, true, true, 4 >, [609](#)
  - Simd128\_impl< true, true, true, 8 >, [618](#)
  - Simd256\_impl< true, false, true, 8 >, [628](#)
  - Simd256\_impl< true, true, false, 2 >, [640](#)
  - Simd256\_impl< true, true, false, 4 >, [656](#)
  - Simd256\_impl< true, true, false, 8 >, [667](#)
  - Simd256\_impl< true, true, true, 2 >, [675](#)
  - Simd256\_impl< true, true, true, 4 >, [686](#), [692](#)
  - Simd256\_impl< true, true, true, 8 >, [701](#)
  - Simd512\_impl< true, false, true, 8 >, [710](#)
  - Simd512\_impl< true, true, false, 8 >, [721](#)
  - Simd512\_impl< true, true, true, 8 >, [729](#)
- mul\_r
  - FieldSimd< \_Field >, [450](#), [451](#)
- mulhi
  - ScalFunctionsBase< Element, typename enable\_if<
    - is\_integral< Element >::value >::type >, [561](#)
 >
    - Simd128\_impl< true, true, false, 2 >, [568](#)
    - Simd128\_impl< true, true, false, 4 >, [578](#)
    - Simd128\_impl< true, true, false, 8 >, [588](#)
    - Simd128\_impl< true, true, true, 2 >, [600](#)
    - Simd128\_impl< true, true, true, 4 >, [609](#)
    - Simd128\_impl< true, true, true, 8 >, [618](#)
    - Simd256\_impl< true, true, false, 2 >, [635](#)
    - Simd256\_impl< true, true, false, 4 >, [647](#), [649](#)
    - Simd256\_impl< true, true, false, 8 >, [664](#)
    - Simd256\_impl< true, true, true, 2 >, [675](#)
    - Simd256\_impl< true, true, true, 4 >, [686](#), [692](#)
    - Simd256\_impl< true, true, true, 8 >, [701](#)
    - Simd512\_impl< true, true, false, 8 >, [717](#)
    - Simd512\_impl< true, true, true, 8 >, [729](#)
- mulhi\_fast
  - Simd128\_impl< true, true, false, 8 >, [593](#)
  - Simd128\_impl< true, true, true, 8 >, [620](#)
  - Simd256\_impl< true, true, false, 8 >, [669](#)
  - Simd256\_impl< true, true, true, 8 >, [703](#)
  - Simd512\_impl< true, true, false, 8 >, [722](#)
  - Simd512\_impl< true, true, true, 8 >, [732](#)
- mulin
  - FieldSimd< \_Field >, [450](#)
  - ScalFunctions< Element >, [556](#)
  - Simd256\_impl< true, false, true, 8 >, [628](#)
  - Simd512\_impl< true, false, true, 8 >, [710](#)
- mullo
  - ScalFunctionsBase< Element, typename enable\_if<
    - is\_integral< Element >::value >::type >, [561](#)
 >
    - Simd128\_impl< true, true, false, 2 >, [572](#)
    - Simd128\_impl< true, true, false, 4 >, [582](#)
    - Simd128\_impl< true, true, false, 8 >, [588](#)
    - Simd128\_impl< true, true, true, 2 >, [600](#)
    - Simd128\_impl< true, true, true, 4 >, [609](#)
    - Simd128\_impl< true, true, true, 8 >, [618](#)
    - Simd256\_impl< true, true, false, 2 >, [640](#)
    - Simd256\_impl< true, true, false, 4 >, [656](#)
    - Simd256\_impl< true, true, false, 8 >, [663](#)
    - Simd256\_impl< true, true, true, 2 >, [675](#)
    - Simd256\_impl< true, true, true, 4 >, [686](#), [692](#)
    - Simd256\_impl< true, true, true, 8 >, [701](#)
    - Simd512\_impl< true, true, false, 8 >, [717](#)
    - Simd512\_impl< true, true, true, 8 >, [729](#)
- mulx
  - ScalFunctionsBase< Element, typename enable\_if<
    - is\_integral< Element >::value >::type >, [561](#)
 >
    - Simd128\_impl< true, true, false, 2 >, [568](#)
    - Simd128\_impl< true, true, false, 4 >, [578](#)
    - Simd128\_impl< true, true, false, 8 >, [588](#)
    - Simd128\_impl< true, true, true, 2 >, [600](#)
    - Simd128\_impl< true, true, true, 4 >, [609](#)
    - Simd128\_impl< true, true, true, 8 >, [618](#)
    - Simd256\_impl< true, true, false, 2 >, [635](#)
    - Simd256\_impl< true, true, false, 4 >, [647](#), [649](#)
    - Simd256\_impl< true, true, false, 8 >, [664](#)
    - Simd256\_impl< true, true, true, 2 >, [676](#)
    - Simd256\_impl< true, true, true, 4 >, [686](#), [692](#)
    - Simd256\_impl< true, true, true, 8 >, [701](#)
    - Simd512\_impl< true, true, false, 8 >, [717](#)
    - Simd512\_impl< true, true, true, 8 >, [729](#)
- mvcnt
  - test-lu.C, [1092](#)
- n
  - Bench< Elt >, [408](#)
  - count\_nonconst\_lvalue\_reference< const T &, O... >, [436](#)
  - count\_nonconst\_lvalue\_reference< T &, O... >, [436](#)
  - count\_nonconst\_lvalue\_reference< T, O... >, [436](#)
  - count\_nonconst\_lvalue\_reference<>, [437](#)
  - Sparse< \_Field, SparseMatrix\_t::COO >, [736](#)
  - Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, [738](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR >, [739](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, [741](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, [743](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL >, [745](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, [746](#)

- Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, 748
- Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, 749
- Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, 751
- Sparse< \_Field, SparseMatrix\_t::SELL >, 753
- Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, 755
- name
  - TestOneMethod< Simd >, 767
- nb\_lref
  - TestOneMethod< Simd >, 766
- nChunks
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, 747
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, 748
  - Sparse< \_Field, SparseMatrix\_t::SELL >, 753
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, 755
- nDenseCols
  - StatsMatrix, 759
- nDenseRows
  - StatsMatrix, 759
- need\_field\_characteristic< Field >, 519
  - value, 519
- need\_field\_characteristic< Givaro::Modular< Field > >, 519
  - value, 520
- need\_field\_characteristic< Givaro::ModularBalanced< Field > >, 520
  - value, 520
- NeedDoublePreAddReduction
  - FFLAS::Protected, 214, 215
- NeedPreAddReduction
  - FFLAS::Protected, 213, 214
- NeedPreApxyReduction
  - FFLAS::Protected, 219, 220
- NeedPreScalReduction
  - FFLAS::Protected, 219
- NeedPreSubReduction
  - FFLAS::Protected, 214
- neg
  - RNSIntegerMod< RNS >, 549
- nElements
  - Sparse< \_Field, SparseMatrix\_t::COO >, 737
  - Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, 738
  - Sparse< \_Field, SparseMatrix\_t::CSR >, 740
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, 741
  - Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, 743
  - Sparse< \_Field, SparseMatrix\_t::ELL >, 745
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, 746
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, 748
  - Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, 750
  - Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, 751
  - Sparse< \_Field, SparseMatrix\_t::SELL >, 753
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, 755
- nEmptyCols
  - StatsMatrix, 759
- nEmptyColsEnd
- StatsMatrix, 759
- nEmptyRows
  - StatsMatrix, 759
- newD
  - FFPACK::Protected, 396
- NEWWINO
  - fgemm\_winograd.inl, 839
- nMOnes
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, 742
  - StatsMatrix, 757
- nnz
  - Sparse< \_Field, SparseMatrix\_t::COO >, 736
  - Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, 738
  - Sparse< \_Field, SparseMatrix\_t::CSR >, 739
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, 741
  - Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, 743
  - Sparse< \_Field, SparseMatrix\_t::ELL >, 745
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, 746
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, 748
  - Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, 750
  - Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, 751
  - Sparse< \_Field, SparseMatrix\_t::SELL >, 753
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, 755
  - StatsMatrix, 757
- none
  - HelperFlag, 474
- nOnes
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, 742
  - StatsMatrix, 757
- nonsquare\_inplace\_v1
  - FFLAS::\_ftranspose\_impl, 190
- nonsquare\_inplace\_v2
  - FFLAS::\_ftranspose\_impl, 190
- NonZeroRandomMatrix
  - FFPACK, 377, 378
- normA
  - MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, 505
  - MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, 507
  - MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeqTrait >, 510
- normB
  - MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, 505
  - MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, 507
  - MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, >,

- ParSeqTrait >, 510
- NORML\_MOD
  - fflas\_simd.h, 872
- NoSimd< T >, 520
  - aligned\_allocator, 521
  - aligned\_vector, 521
  - alignment, 521
  - compliant, 521
  - is\_same\_element, 521
  - scalar\_t, 520
  - type\_string, 521
  - valid, 521
  - vect\_size, 521
  - vect\_t, 520
- NoSimdSparseMatrix
  - FFLAS, 73
- NOSPLIT
  - parallel.h, 1038
- not\_inplace
  - FFLAS::\_franspose\_impl, 190
- nOthers
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, 742
  - StatsMatrix, 757
- NotMKLSparseMatrixFormat
  - FFLAS, 73
- NotZOSparseMatrix
  - FFLAS, 72
- NullSpaceBasis
  - FFPACK, 329, 373
- NullSpaceBasis\_modular\_double
  - ffpack.C, 1000
  - ffpack\_c.h, 1020
- NUM\_THREADS
  - parallel.h, 1034
- NUMARGS
  - parallel.h, 1036
- number\_kind
  - FFLAS, 76
- numBlock
  - ForStrategy1D< blocksize\_t, Cut, Param >, 463
- numblocks
  - ForStrategy1D< blocksize\_t, Cut, Param >, 462
- numColBlock
  - ForStrategy2D< blocksize\_t, Cut, Param >, 466
- numRowBlock
  - ForStrategy2D< blocksize\_t, Cut, Param >, 466
- numthreads
  - Parallel< C, P >, 522
  - Sequential, 563
- one
  - FFLAS, 76
  - RNSInteger< RNS >, 545
  - RNSIntegerMod< RNS >, 552
  - Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, 751
- OPENBLAS\_NUM\_THREADS
  - config.h, 819
- operator!=
  - rns\_double\_elt\_cstptr, 535
  - rns\_double\_elt\_ptr, 537
- operator<
  - rns\_double\_elt\_cstptr, 535
  - rns\_double\_elt\_ptr, 537
- operator<<
  - Compose< H1, H2 >, 425
  - FFLAS, 151
  - ForStrategy2D< blocksize\_t, Cut, Param >, 465
  - MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, 505
  - MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, 507
  - MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< Dest >, ParSeqTrait >, 508
  - MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeqTrait >, 510
  - MMHelper< Field, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, 512
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, 502
  - Parallel< C, P >, 522
  - Sequential, 564
  - test-fsytrf.C, 1077
  - test-simd.C, 1102
- operator()
  - callLUdivine\_small< double >, 411
  - callLUdivine\_small< Element >, 411
  - callLUdivine\_small< float >, 412
  - Failure, 444, 445
  - readMyMachineType< Field, mpz\_t >, 526
  - readMyMachineType< Field, T >, 525
  - RNSInteger< RNS >::RandIter, 523
  - RNSIntegerMod< RNS >::RandIter, 524
  - rnsRandIter< RNS >, 553
  - ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >::FloatingPointTestDistribution, 460
  - tfn\_minus, 767
  - tfn\_minus\_eq, 768
  - tfn\_mul, 768
  - tfn\_mul\_eq, 768
  - tfn\_plus, 769
  - tfn\_plus\_eq, 769
- operator+
  - rns\_double\_elt\_cstptr, 534
  - rns\_double\_elt\_ptr, 537
- operator++
  - ForStrategy1D< blocksize\_t, Cut, Param >, 462
  - ForStrategy2D< blocksize\_t, Cut, Param >, 464
  - rns\_double\_elt\_cstptr, 534
  - rns\_double\_elt\_ptr, 537
- operator+=
  - rns\_double\_elt\_cstptr, 534
  - rns\_double\_elt\_ptr, 537

- operator-
  - rns\_double\_elt\_cstptr, [534](#)
  - rns\_double\_elt\_ptr, [537](#)
- operator--
  - rns\_double\_elt\_cstptr, [534](#)
  - rns\_double\_elt\_ptr, [537](#)
- operator-=
  - rns\_double\_elt\_cstptr, [534](#)
  - rns\_double\_elt\_ptr, [537](#)
- operator=
  - Coo< Field >, [433](#)
  - Coo< ValT, ldxT >, [431](#), [434](#)
  - FieldSimd< \_Field >, [448](#)
  - Info, [478](#), [480](#)
  - rns\_double\_elt\_cstptr, [534](#)
  - rns\_double\_elt\_ptr, [537](#)
- operator&
  - rns\_double\_elt, [532](#)
  - rns\_double\_elt\_cstptr, [534](#), [535](#)
  - rns\_double\_elt\_ptr, [536](#), [537](#)
- operator[]
  - rns\_double\_elt\_cstptr, [534](#)
  - rns\_double\_elt\_ptr, [536](#), [537](#)
- operator\*
  - rns\_double\_elt\_cstptr, [534](#)
  - rns\_double\_elt\_ptr, [536](#)
- other
  - FFLAS, [76](#)
  - rns\_double\_elt\_cstptr, [535](#)
  - rns\_double\_elt\_ptr, [538](#)
- Outmax
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [503](#)
- Outmin
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [503](#)
- outputs\_scalar
  - TestOneMethod< Simd >, [767](#)
- outputs\_simd
  - TestOneMethod< Simd >, [767](#)
- p
  - HelperMod< Field, ElementCategories::MachineIntTag >, [475](#)
  - HelperMod< Field, FFLAS::ElementCategories::ArbitraryPrecIntTag >, [476](#)
  - HelperMod< Field, FFLAS::ElementCategories::FixedPrecIntTag >, [477](#)
  - HelperMod< Field, FFLAS::ElementCategories::MachineFPIntTag >, [477](#)
- pack
  - ScalFunctions< Element >, [558](#)
  - Simd128\_impl< true, true, false, 2 >, [571](#)
  - Simd128\_impl< true, true, false, 4 >, [581](#)
  - Simd128\_impl< true, true, false, 8 >, [591](#)
  - Simd128\_impl< true, true, true, 2 >, [599](#)
  - Simd128\_impl< true, true, true, 4 >, [608](#)
  - Simd128\_impl< true, true, true, 8 >, [617](#)
  - Simd256\_impl< true, false, true, 8 >, [627](#)
  - Simd256\_impl< true, true, false, 2 >, [638](#)
  - Simd256\_impl< true, true, false, 4 >, [654](#)
  - Simd256\_impl< true, true, false, 8 >, [667](#)
  - Simd256\_impl< true, true, true, 2 >, [674](#)
  - Simd256\_impl< true, true, true, 4 >, [685](#), [691](#)
  - Simd256\_impl< true, true, true, 8 >, [700](#)
  - Simd512\_impl< true, false, true, 8 >, [709](#)
  - Simd512\_impl< true, true, false, 8 >, [720](#)
  - Simd512\_impl< true, true, true, 8 >, [728](#)
- Simd256\_impl< true, true, false, 2 >, [638](#)
- Simd256\_impl< true, true, false, 4 >, [654](#)
- Simd256\_impl< true, true, false, 8 >, [667](#)
- Simd256\_impl< true, true, true, 2 >, [674](#)
- Simd256\_impl< true, true, true, 4 >, [685](#), [691](#)
- Simd256\_impl< true, true, true, 8 >, [700](#)
- Simd512\_impl< true, false, true, 8 >, [709](#)
- Simd512\_impl< true, true, false, 8 >, [720](#)
- Simd512\_impl< true, true, true, 8 >, [728](#)
- pack\_even
  - ScalFunctions< Element >, [558](#)
  - Simd128\_impl< true, true, false, 2 >, [571](#)
  - Simd128\_impl< true, true, false, 4 >, [581](#)
  - Simd128\_impl< true, true, false, 8 >, [591](#)
  - Simd128\_impl< true, true, true, 2 >, [599](#)
  - Simd128\_impl< true, true, true, 4 >, [608](#)
  - Simd128\_impl< true, true, true, 8 >, [617](#)
  - Simd256\_impl< true, false, true, 8 >, [627](#)
  - Simd256\_impl< true, true, false, 2 >, [638](#)
  - Simd256\_impl< true, true, false, 4 >, [654](#)
  - Simd256\_impl< true, true, false, 8 >, [666](#)
  - Simd256\_impl< true, true, true, 2 >, [674](#)
  - Simd256\_impl< true, true, true, 4 >, [684](#), [691](#)
  - Simd256\_impl< true, true, true, 8 >, [700](#)
  - Simd512\_impl< true, false, true, 8 >, [709](#)
  - Simd512\_impl< true, true, false, 8 >, [720](#)
  - Simd512\_impl< true, true, true, 8 >, [728](#)
- pack\_lhs
  - FFLAS::details, [205](#)
- pack\_odd
  - ScalFunctions< Element >, [558](#)
  - Simd128\_impl< true, true, false, 2 >, [571](#)
  - Simd128\_impl< true, true, false, 4 >, [581](#)
  - Simd128\_impl< true, true, false, 8 >, [591](#)
  - Simd128\_impl< true, true, true, 2 >, [599](#)
  - Simd128\_impl< true, true, true, 4 >, [608](#)
  - Simd128\_impl< true, true, true, 8 >, [617](#)
  - Simd256\_impl< true, false, true, 8 >, [627](#)
  - Simd256\_impl< true, true, false, 2 >, [638](#)
  - Simd256\_impl< true, true, false, 4 >, [654](#)
  - Simd256\_impl< true, true, false, 8 >, [667](#)
  - Simd256\_impl< true, true, true, 2 >, [674](#)
  - Simd256\_impl< true, true, true, 4 >, [685](#), [691](#)
  - Simd256\_impl< true, true, true, 8 >, [700](#)
  - Simd512\_impl< true, false, true, 8 >, [709](#)
  - Simd512\_impl< true, true, false, 8 >, [720](#)
  - Simd512\_impl< true, true, true, 8 >, [728](#)
- pack\_rhs
  - FFLAS::details, [205](#)
- PACKAGE
  - config.h, [819](#)
- PACKAGE\_BUGREPORT
  - config.h, [819](#)
- PACKAGE\_NAME
  - config.h, [819](#)
- PACKAGE\_STRING
  - config.h, [819](#)
- PACKAGE\_TARNAME

- config.h, [819](#)
- PACKAGE\_URL
  - config.h, [819](#)
- PACKAGE\_VERSION
  - config.h, [819](#)
- PAR\_BLOCK
  - parallel.h, [1033](#)
- Parallel
  - Parallel< C, P >, [522](#)
- Parallel< C, P >, [522](#)
  - Cut, [522](#)
  - numthreads, [522](#)
  - operator<<, [522](#)
  - Parallel, [522](#)
  - Param, [522](#)
  - set\_numthreads, [522](#)
- parallel.h, [1032](#)
  - \_\_FFLASFFPACK\_SEQUENTIAL, [1033](#)
  - BARRIER, [1033](#)
  - BEGIN\_PARALLEL\_MAIN, [1034](#)
  - CHECK\_DEPENDENCIES, [1033](#)
  - COMMA, [1036](#)
  - CONSTREFERENCE, [1034](#)
  - END\_PARALLEL\_MAIN, [1034](#)
  - FOR1D, [1035](#)
  - FOR2D, [1035](#)
  - FORBLOCK1D, [1034](#)
  - FORBLOCK2D, [1035](#)
  - index\_t, [1033](#)
  - MAX\_THREADS, [1034](#)
  - MODE, [1036](#)
  - NOSPLIT, [1038](#)
  - NUM\_THREADS, [1034](#)
  - NUMARGS, [1036](#)
  - PAR\_BLOCK, [1033](#)
  - PARFOR1D, [1035](#)
  - PARFOR2D, [1036](#)
  - PARFORBLOCK1D, [1035](#)
  - PARFORBLOCK2D, [1036](#)
  - PP\_ARG\_N, [1037](#)
  - PP\_NARG\_, [1036](#)
  - PP\_RSEQ\_N, [1038](#)
  - READ, [1034](#)
  - READWRITE, [1034](#)
  - RETURNPARAM, [1036](#)
  - SET\_THREADS, [1034](#)
  - splitt, [1039](#)
  - SPLITTER, [1039](#)
  - splitting\_0, [1038](#)
  - splitting\_1, [1038](#)
  - splitting\_2, [1038](#)
  - splitting\_3, [1038](#)
  - SYNCH\_GROUP, [1033](#)
  - TASK, [1033](#)
  - THREAD\_INDEX, [1033](#)
  - VALUE, [1034](#)
  - WAIT, [1033](#)
  - WRITE, [1034](#)
- Param
  - Parallel< C, P >, [522](#)
- PARFOR1D
  - parallel.h, [1035](#)
- PARFOR2D
  - parallel.h, [1036](#)
- PARFORBLOCK1D
  - parallel.h, [1035](#)
- PARFORBLOCK2D
  - parallel.h, [1036](#)
- parseArguments
  - FFLAS, [185](#)
- parseq
  - MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, [505](#)
  - MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, [507](#)
  - MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< Dest >, ParSeqTrait >, [508](#)
  - MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeqTrait >, [510](#)
  - MMHelper< Field, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, [512](#)
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, [503](#)
  - TRSMHelper< ReclterTrait, ParSeqTrait >, [771](#)
- PBASECASE\_K
  - ffpack\_ppluq.inl, [943](#)
- pColumnEchelonForm
  - FFPACK, [316](#)
- pColumnEchelonForm\_modular\_double
  - ffpack.C, [995](#)
- pColumnEchelonForm\_modular\_float
  - ffpack.C, [996](#)
- pColumnEchelonForm\_modular\_int32\_t
  - ffpack.C, [997](#)
- pColumnRankProfile
  - FFPACK, [332](#)
- pDet
  - FFPACK, [327](#)
- perm
  - Info, [479](#), [480](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL >, [753](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, [755](#)
- PermApplyS
  - FFPACK, [358](#)
- PermApplyS\_double
  - ffpack.C, [988](#)
  - ffpack\_c.h, [1010](#)
- PermApplyT
  - FFPACK, [360](#)
- PermApplyT\_double
  - ffpack.C, [989](#)
  - ffpack\_c.h, [1010](#)
- pfadd



- FFLAS, [79](#)
- pfaddin
  - FFLAS, [79](#)
- pfgemm
  - FFLAS, [142](#), [182–184](#)
- pfgemm\_1D\_rec
  - FFLAS, [142](#)
- pfgemm\_2D\_rec
  - FFLAS, [143](#)
- pfgemm\_3D\_rec
  - FFLAS, [143](#)
- pfgemm\_3D\_rec2
  - FFLAS, [144](#)
- pfgemm\_variants.inl, [1039](#)
- pfgemv.inl, [1040](#)
- pfrand
  - FFLAS, [181](#)
- pfreduce
  - FFLAS, [109](#)
- pfspmm
  - FFLAS::sparse\_details, [233–235](#)
  - FFLAS::sparse\_details\_impl, [250](#), [257](#), [258](#), [260](#), [261](#), [271](#), [272](#)
- pfspmm\_dispatch
  - FFLAS::sparse\_details, [232](#)
- pfspmm\_mone
  - FFLAS::sparse\_details\_impl, [251](#)
- pfspmm\_one
  - FFLAS::sparse\_details\_impl, [250](#), [251](#)
- pfspmm\_zo
  - FFLAS::sparse\_details\_impl, [262](#)
- pfspmv
  - FFLAS::sparse\_details, [235](#), [236](#)
  - FFLAS::sparse\_details\_impl, [251](#), [252](#), [258](#), [259](#), [262](#), [268](#), [272](#), [274](#)
- pfspmv\_mone
  - FFLAS::sparse\_details\_impl, [252](#), [253](#), [263](#), [268](#), [269](#), [275](#)
- pfspmv\_one
  - FFLAS::sparse\_details\_impl, [252](#), [253](#), [263](#), [268](#), [269](#), [274](#), [275](#)
- pfspmv\_task
  - FFLAS::sparse\_details\_impl, [252](#)
- pfsub
  - FFLAS, [79](#)
- pfsubin
  - FFLAS, [80](#)
- pfzero
  - FFLAS, [181](#)
- PLUQ
  - FFPACK, [313–315](#), [363](#), [364](#), [367](#)
- pluq.C, [777](#), [778](#)
- CUBE, [777](#)
- GFOPS, [777](#)
- main, [777](#), [778](#)
- PLUQ\_basecaseCrout
  - FFPACK, [362](#)
- PLUQ\_basecaseV2
  - FFPACK, [362](#)
- PLUQ\_basecaseV3
  - FFPACK, [362](#)
- PLUQ\_modular\_double
  - ffpack.C, [992](#)
  - ffpack\_c.h, [1013](#)
- PLUQtoEchelonPermutation
  - FFPACK, [341](#)
  - ffpack.C, [1004](#)
  - ffpack\_c.h, [1024](#)
- pm1
  - HelperFlag, [474](#)
- pMMH
  - TRSMHelper< ReclterTrait, ParSeqTrait >, [771](#)
- pow50rem
  - HelperMod< Field, ElementCategories::MachineIntTag >, [475](#)
- PP\_ARG\_N
  - parallel.h, [1037](#)
- PP\_NARG\_
  - parallel.h, [1036](#)
- PP\_RSEQ\_N
  - parallel.h, [1038](#)
- pPLUQ
  - FFPACK, [314](#)
- pRank
  - FFPACK, [326](#)
- preamble
  - FFLAS, [186](#)
- precompute\_cst
  - rns\_double, [528](#)
  - rns\_double\_extended, [540](#)
- pReducedColumnEchelonForm
  - FFPACK, [319](#)
- pReducedColumnEchelonForm\_modular\_double
  - ffpack.C, [996](#)
- pReducedColumnEchelonForm\_modular\_float
  - ffpack.C, [996](#)
- pReducedColumnEchelonForm\_modular\_int32\_t
  - ffpack.C, [997](#)
- pReducedRowEchelonForm
  - FFPACK, [320](#)
- pReducedRowEchelonForm\_modular\_double
  - ffpack.C, [996](#)
- pReducedRowEchelonForm\_modular\_float
  - ffpack.C, [997](#)
- pReducedRowEchelonForm\_modular\_int32\_t
  - ffpack.C, [998](#)
- prefetch
  - FFLAS, [189](#)
- print
  - Failure, [445](#)
- printHelpMessage
  - args-parser.h, [1042](#)
- printPolynomial
  - test-charpoly-check.C, [1055](#)
- printvect
  - test-compressQ.C, [1056](#)

- productBruhatxTS
  - FFPACK, [346](#), [349](#)
- pRowEchelonForm
  - FFPACK, [318](#)
- pRowEchelonForm\_modular\_double
  - ffpack.C, [995](#)
- pRowEchelonForm\_modular\_float
  - ffpack.C, [996](#)
- pRowEchelonForm\_modular\_int32\_t
  - ffpack.C, [997](#)
- pRowRankProfile
  - FFPACK, [331](#)
- PSeq
  - benchmark-fgemm-rns.C, [789](#)
- pSolve
  - FFPACK, [329](#)
- PTRSM\_HYBRID\_THRESHOLD
  - fflas\_pfrsm.inl, [871](#)
- PURE
  - fflas\_simd.h, [872](#)
- queryCacheSizes
  - FFLAS, [189](#)
- queryL1CacheSize
  - FFLAS, [189](#)
- queryTopLevelCacheSize
  - FFLAS, [189](#)
- RandInt
  - FFPACK, [381](#)
- RandIter
  - RNSInteger< RNS >::RandIter, [523](#)
  - RNSIntegerMod< RNS >::RandIter, [524](#)
- random
  - RNSInteger< RNS >::RandIter, [523](#)
  - RNSIntegerMod< RNS >::RandIter, [524](#)
  - rnsRandIter< RNS >, [553](#)
- RandomIndexSubset
  - FFPACK, [382](#)
- RandomKrylovPrecond
  - FFPACK::Protected, [395](#)
- RandomLTQSMatixWithRankandQSorder
  - FFPACK, [390](#)
- RandomLTQSRankProfileMatrix
  - FFPACK, [384](#)
- RandomMatrix
  - FFPACK, [378](#), [379](#)
- RandomMatrixWithDet
  - FFPACK, [389](#)
- RandomMatrixWithRank
  - FFPACK, [381](#), [382](#)
- RandomMatrixWithRankandRandomRPM
  - FFPACK, [387](#)
- RandomMatrixWithRankandRPM
  - FFPACK, [384](#), [385](#)
- RandomNullSpaceVector
  - FFPACK, [329](#), [347](#), [373](#)
- RandomNullSpaceVector\_modular\_double
  - ffpack.C, [1000](#)
- ffpack\_c.h, [1020](#)
- RandomPermutation
  - FFPACK, [383](#)
- RandomRankProfileMatrix
  - FFPACK, [383](#)
- RandomSymmetricMatrix
  - FFPACK, [381](#)
- RandomSymmetricMatrixWithRankandRandomRPM
  - FFPACK, [388](#)
- RandomSymmetricMatrixWithRankandRPM
  - FFPACK, [385](#), [386](#)
- RandomSymmetricRankProfileMatrix
  - FFPACK, [384](#)
- RandomTriangularMatrix
  - FFPACK, [379](#), [380](#)
- Rank
  - FFPACK, [325](#), [326](#), [371](#)
- rank.C, [805](#)
  - main, [805](#)
- Rank\_modular\_double
  - ffpack.C, [999](#)
  - ffpack\_c.h, [1019](#)
- RankProfileFromLU
  - FFPACK, [332](#)
  - ffpack.C, [1001](#)
  - ffpack\_c.h, [1021](#)
- READ
  - parallel.h, [1034](#)
- read\_field
  - Matio.h, [1049](#)
- read\_sparse.h, [908](#)
  - DNS\_BIN\_VER, [909](#)
  - mask\_t, [909](#)
- readDnsFormat
  - FFLAS, [152](#)
- readMachineType
  - FFLAS, [152](#)
- ReadMatrix
  - FFLAS, [186](#)
- readMyMachineType< Field, mpz\_t >, [525](#)
  - Element, [526](#)
  - Element\_ptr, [526](#)
  - operator(), [526](#)
- readMyMachineType< Field, T >, [525](#)
  - Element, [525](#)
  - Element\_ptr, [525](#)
  - operator(), [525](#)
- readOrRandomMatrixWithRankAndRandomRPM
  - test-nullspace.C, [1094](#)
- readSmsFormat
  - FFLAS, [151](#)
- readSprFormat
  - FFLAS, [151](#)
- READWRITE
  - parallel.h, [1034](#)
- Rec\_Initialize
  - benchmark-pluq.C, [801](#)
- RecInt, [398](#)



- recLevel
  - MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, 505
  - MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, 507
  - MMHelper< Field, AlgoTrait, ModeCategories::ConversionTag, Dest >, ParSeqTrait >, 508
  - MMHelper< Field, AlgoTrait, ModeCategories::ConversionTag, ElementCategories::RNSElementTag >, ParSeqTrait >, 510
  - MMHelper< Field, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, 512
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, 502
- Recursive, 526
- reduce
  - FFLAS::vectorised, 281–283
  - rns\_double, 529
  - rns\_double\_extended, 540
  - RNSInteger< RNS >, 544
  - RNSIntegerMod< RNS >, 549
- reduce\_modp
  - RNSIntegerMod< RNS >, 550, 551
- reduce\_modp\_rnsmajor
  - RNSIntegerMod< RNS >, 551
- ReducedColumnEchelonForm
  - FFPACK, 318, 319, 369
- ReducedColumnEchelonForm\_modular\_double
  - ffpack.C, 993
  - ffpack\_c.h, 1016
- ReducedColumnEchelonForm\_modular\_float
  - ffpack.C, 994
  - ffpack\_c.h, 1016
- ReducedColumnEchelonForm\_modular\_int32\_t
  - ffpack.C, 995
  - ffpack\_c.h, 1017
- ReducedRowEchelonForm
  - FFPACK, 319, 320, 368
- ReducedRowEchelonForm2\_modular\_double
  - ffpack\_c.h, 1017
- ReducedRowEchelonForm\_modular\_double
  - ffpack.C, 993
  - ffpack\_c.h, 1016
- ReducedRowEchelonForm\_modular\_float
  - ffpack.C, 994
  - ffpack\_c.h, 1016
- ReducedRowEchelonForm\_modular\_int32\_t
  - ffpack.C, 995
  - ffpack\_c.h, 1017
- REF\_modular\_double
  - ffpack\_c.h, 1017
- regression-check.C, 1053
  - check1, 1053
  - check2, 1053
  - check3, 1054
  - check4, 1054
  - checkZeroDimCharpoly, 1054
  - checkZeroDimMinPoly, 1054
  - gf2ModularBalanced, 1054
  - main, 1054
- Residu
  - Bench< Elt >, 407
  - Test< Elt >, 762
- RETURNPARAM
  - parallel.h, 1036
- Ring
  - CheckerImplem\_charpoly< Givaro::ZRing< Givaro::Integer >, Polynomial >, 414
  - RNSInteger< RNS >::Randlter, 523
  - RNSIntegerMod< RNS >::Randlter, 524
  - rnsRandlter< RNS >, 553
- rint< K >, 526
- RNS, 43
  - benchmark-fgemm-rns.C, 788
- rns
  - RNSInteger< RNS >, 543
  - RNSIntegerMod< RNS >, 547
- rns-double-elt.h, 947
- rns-double-recint.inl, 947
  - \_\_FFLASFFPACK\_field\_rns\_double\_recint\_INL, 948
- rns-double.h, 948
  - ROUND\_DOWN, 948
- rns-double.inl, 949
  - \_\_FFLASFFPACK\_field\_rns\_double\_INL, 949
- rns-integer-mod.h, 949
- rns-integer.h, 950
- rns.h, 950
- rns.inl, 951
  - \_\_FFLASFFPACK\_field\_rns\_INL, 951
- rns\_double, 526
  - \_M, 530
  - \_MMi, 531
  - \_Mi, 530
  - \_basis, 530
  - \_basisMax, 530
  - \_crt\_in, 531
  - \_crt\_out, 531
  - \_field\_rns, 530
  - \_invbasis, 530
  - \_ldm, 531
  - \_mi\_sum, 531
  - \_negbasis, 530
  - \_pbits, 531
  - \_size, 531
- BasisElement, 528
- ConstElement\_ptr, 528
- convert, 529, 530
- convert\_transpose, 529
- Element, 528
- Element\_ptr, 528
- init, 528–530
- init\_transpose, 529

- integer, 527
- ModField, 527
- precompute\_cst, 528
- reduce, 529
- rns\_double, 528
- rns\_double\_elt, 531
  - \_alloc, 532
  - \_ptr, 532
  - \_stride, 532
  - ~rns\_double\_elt, 532
  - operator&, 532
  - rns\_double\_elt, 532
- rns\_double\_elt\_cstptr, 533
  - \_alloc, 535
  - \_ptr, 535
  - \_stride, 535
  - operator!=, 535
  - operator<, 535
  - operator+, 534
  - operator++, 534
  - operator+=", 534
  - operator-, 534
  - operator--, 534
  - operator=, 534
  - operator=, 534
  - operator&, 534, 535
  - operator[], 534
  - operator\*, 534
  - other, 535
  - rns\_double\_elt\_cstptr, 533, 534
- rns\_double\_elt\_ptr, 535
  - \_alloc, 538
  - \_ptr, 538
  - \_stride, 538
  - operator!=, 537
  - operator<, 537
  - operator+, 537
  - operator++, 537
  - operator+=", 537
  - operator-, 537
  - operator--, 537
  - operator=, 537
  - operator=, 537
  - operator&, 536, 537
  - operator[], 536, 537
  - operator\*, 536
  - other, 538
  - rns\_double\_elt\_ptr, 536
- rns\_double\_extended, 538
  - \_M, 541
  - \_MMi, 541
  - \_Mi, 541
  - \_basis, 541
  - \_basisMax, 541
  - \_crt\_in, 541
  - \_crt\_out, 541
  - \_field\_rns, 541
  - \_invbasis, 541
  - \_ldm, 541
  - \_negbasis, 541
  - \_pbits, 541
  - \_size, 541
  - BasisElement, 539
  - ConstElement\_ptr, 539
  - convert, 540
  - Element, 539
  - Element\_ptr, 539
  - init, 540
  - integer, 539
  - ModField, 539
  - precompute\_cst, 540
  - reduce, 540
  - rns\_double\_extended, 539
- RNSElementTag, 542
- RNSInteger
  - RNSInteger< RNS >, 543
- RNSInteger< RNS >, 542
  - \_rns, 545
  - assign, 545
  - BasisElement, 543
  - cardinality, 544
  - characteristic, 544
  - ConstElement\_ptr, 543
  - convert, 544
  - Element, 543
  - Element\_ptr, 543
  - init, 544
  - integer, 543
  - isMOne, 544
  - isOne, 543
  - isZero, 544
  - mOne, 545
  - one, 545
  - reduce, 544
  - rns, 543
  - RNSInteger, 543
  - size, 543
  - write, 545
  - zero, 545
- RNSInteger< RNS >::RandIter, 523
  - operator(), 523
  - RandIter, 523
  - random, 523
  - ring, 523
- RNSIntegerMod
  - RNSIntegerMod< RNS >, 547
- RNSIntegerMod< RNS >, 545
  - \_F, 551
  - \_Mi\_modp\_rns, 551
  - \_RNSdelayed, 552
  - \_iM\_modp\_rns, 551
  - \_p, 551
  - \_rns, 551
  - add, 549
  - areEqual, 550
  - assign, 549

- axpyin, [550](#)
- BasisElement, [547](#)
- cardinality, [548](#)
- characteristic, [548](#)
- ConstElement\_ptr, [547](#)
- convert, [549](#)
- delayed, [547](#)
- Element, [547](#)
- Element\_ptr, [547](#)
- init, [548](#), [549](#)
- integer, [547](#)
- inv, [550](#)
- isMOne, [548](#)
- isOne, [548](#)
- isZero, [548](#)
- maxElement, [548](#)
- minElement, [548](#)
- ModField, [547](#)
- mOne, [552](#)
- mul, [550](#)
- neg, [549](#)
- one, [552](#)
- reduce, [549](#)
- reduce\_modp, [550](#), [551](#)
- reduce\_modp\_rnsmajor, [551](#)
- rns, [547](#)
- RNSIntegerMod, [547](#)
- size, [547](#)
- sub, [549](#)
- write, [550](#)
- write\_matrix, [550](#)
- write\_matrix\_long, [551](#)
- zero, [552](#)
- RNSIntegerMod< RNS >::RandIter, [524](#)
  - operator(), [524](#)
  - RandIter, [524](#)
  - random, [524](#)
  - ring, [524](#)
- RNSModulus
  - FFLAS::CuttingStrategy, [198](#)
- rnsRandIter
  - rnsRandIter< RNS >, [552](#)
- rnsRandIter< RNS >, [552](#)
  - operator(), [553](#)
  - random, [553](#)
  - ring, [553](#)
  - rnsRandIter, [552](#)
- round
  - ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >, [559](#)
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, [561](#)
  - Simd128\_impl< true, true, false, 2 >, [573](#)
  - Simd128\_impl< true, true, false, 4 >, [583](#)
  - Simd128\_impl< true, true, false, 8 >, [593](#)
  - Simd128\_impl< true, true, true, 2 >, [602](#)
  - Simd128\_impl< true, true, true, 4 >, [611](#)
  - Simd128\_impl< true, true, true, 8 >, [620](#)
  - Simd256\_impl< true, false, true, 8 >, [630](#)
  - Simd256\_impl< true, true, false, 2 >, [641](#)
  - Simd256\_impl< true, true, false, 4 >, [658](#)
  - Simd256\_impl< true, true, false, 8 >, [668](#)
  - Simd256\_impl< true, true, true, 2 >, [678](#)
  - Simd256\_impl< true, true, true, 4 >, [688](#), [694](#)
  - Simd256\_impl< true, true, true, 8 >, [703](#)
  - Simd512\_impl< true, false, true, 8 >, [712](#)
  - Simd512\_impl< true, true, false, 8 >, [722](#)
  - Simd512\_impl< true, true, true, 8 >, [732](#)
- ROUND\_DOWN
  - fflas\_sparse.h, [885](#)
  - rns-double.h, [948](#)
- Row, [553](#)
- row
  - Coo< Field >, [433](#)
  - Coo< ValT, IdxT >, [431](#), [434](#)
  - Sparse< \_Field, SparseMatrix\_t::COO >, [736](#)
  - Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, [738](#)
- rowblockindex
  - ForStrategy2D< blocksize\_t, Cut, Param >, [465](#)
- rowBlockSize
  - ForStrategy2D< blocksize\_t, Cut, Param >, [466](#)
- rowdim
  - StatsMatrix, [757](#)
- RowEchelonForm
  - FFPACK, [317](#), [318](#), [368](#)
- RowEchelonForm\_modular\_double
  - ffpack.C, [992](#)
  - ffpack\_c.h, [1014](#)
- RowEchelonForm\_modular\_float
  - ffpack.C, [993](#)
  - ffpack\_c.h, [1015](#)
- RowEchelonForm\_modular\_int32\_t
  - ffpack.C, [994](#)
  - ffpack\_c.h, [1015](#)
- rownumblocks
  - ForStrategy2D< blocksize\_t, Cut, Param >, [464](#)
- RowRankProfile
  - FFPACK, [330](#), [331](#), [373](#)
- RowRankProfile\_modular\_double
  - ffpack.C, [1001](#)
  - ffpack\_c.h, [1020](#)
- RowRankProfileSubmatrix
  - FFPACK, [334](#), [374](#)
- RowRankProfileSubmatrix\_modular\_double
  - ffpack.C, [1002](#)
  - ffpack\_c.h, [1021](#)
- RowRankProfileSubmatrixIndices
  - FFPACK, [333](#), [374](#)
- RowRankProfileSubmatrixIndices\_modular\_double
  - ffpack.C, [1001](#)
  - ffpack\_c.h, [1021](#)
- ruint< K >, [553](#)
- run
  - Bench< Elt >, [408](#)
  - Test< Elt >, [763](#)

- run\_with\_field
  - benchmark-charpoly.C, 780
  - benchmark-fdot.C, 786
  - benchmark-quasisep.C, 802
  - test-charpoly.C, 1055
  - test-echelon.C, 1060
  - test-fdot.C, 1062
  - test-fgemm-check.C, 1063
  - test-fgemm.C, 1066
  - test-fgemv.C, 1067
  - test-fger.C, 1069
  - test-fgesv.C, 1070
  - test-finit.C, 1071
  - test-fsyr2k.C, 1074
  - test-fsyrk.C, 1076
  - test-fsytrf.C, 1077
  - test-ftrmm.C, 1079
  - test-ftrmv.C, 1080
  - test-ftrsm.C, 1081
  - test-ftrssyr2k.C, 1083
  - test-ftrstr.C, 1084
  - test-ftrsv.C, 1085
  - test-ftrtri.C, 1086
  - test-io.C, 1088
  - test-lu.C, 1091
  - test-minpoly.C, 1093
  - test-nullspace.C, 1095
  - test-quasisep.C, 1098
  - test-rankprofiles.C, 1099
  - test-solve.C, 1103
- run\_with\_Integer
  - test-fdot.C, 1062
- saxpy\_
  - config-blas.h, 812
- ScalAndReduce
  - FFLAS::Protected, 215, 219
- scalar\_t
  - FieldSimd< \_Field >, 447
  - NoSimd< T >, 520
  - Simd128\_impl< true, true, false, 2 >, 566
  - Simd128\_impl< true, true, false, 4 >, 576
  - Simd128\_impl< true, true, false, 8 >, 586
  - Simd128\_impl< true, true, true, 2 >, 596
  - Simd128\_impl< true, true, true, 4 >, 605
  - Simd128\_impl< true, true, true, 8 >, 614
  - Simd256\_impl< true, false, true, 8 >, 625
  - Simd256\_impl< true, true, false, 2 >, 633
  - Simd256\_impl< true, true, false, 4 >, 645
  - Simd256\_impl< true, true, false, 8 >, 662
  - Simd256\_impl< true, true, true, 2 >, 671
  - Simd256\_impl< true, true, true, 4 >, 682
  - Simd256\_impl< true, true, true, 8 >, 697
  - Simd512\_impl< true, false, true, 8 >, 707
  - Simd512\_impl< true, true, false, 8 >, 715
  - Simd512\_impl< true, true, true, 8 >, 725
- ScalFunctions< Element >, 553
  - add, 555
  - addin, 555
  - blend, 558
  - div, 556
  - eq, 557
  - fmadd, 556
  - fmaddin, 556
  - fmsub, 556
  - fmsubin, 556
  - fnmadd, 556
  - fnmaddin, 557
  - genInputs, 554
  - genInputsWithZero, 554
  - greater, 557
  - greater\_eq, 557
  - lesser, 557
  - lesser\_eq, 557
  - mul, 556
  - mulin, 556
  - pack, 558
  - pack\_even, 558
  - pack\_odd, 558
  - sub, 555
  - subin, 555
  - unpackhi, 557
  - unpacklo, 557
  - unpacklohi, 558
  - vand, 555
  - vandnot, 555
  - vectElt, 554
  - vor, 555
  - vxor, 555
  - zero, 555
- ScalFunctionsBase< Element, Enable >, 558
- ScalFunctionsBase< Element, typename enable\_if<
  - is\_floating\_point< Element >::value >::type
  - >, 559
  - \_zero, 560
  - blendv, 559
  - ceil, 559
  - cmp\_false, 560
  - cmp\_true, 560
  - floor, 559
  - fma, 560
  - get\_default\_random\_generator, 559
  - round, 559
- ScalFunctionsBase< Element, typename enable\_if<
  - is\_floating\_point< Element >::value >::type
  - >::FloatingPointTestDistribution, 460
  - FloatingPointTestDistribution, 460
  - IntType, 460
  - operator(), 460
- ScalFunctionsBase< Element, typename enable\_if<
  - is\_integral< Element >::value >::type >, 560
  - \_zero, 563
  - cmp\_false, 563
  - cmp\_true, 563
  - fma, 561
  - fmaddx, 561
  - fmaddxin, 561

- fmsubx, [562](#)
- fmsubxin, [562](#)
- fnmaddx, [562](#)
- fnmaddxin, [562](#)
- get\_default\_random\_generator, [561](#)
- mulhi, [561](#)
- mullo, [561](#)
- mulx, [561](#)
- round, [561](#)
- sll, [562](#)
- sra, [562](#)
- srl, [562](#)
- scalp
  - FFLAS::vectorised, [283](#), [284](#)
  - FFLAS::vectorised::unswitch, [286](#)
- schedule\_bini.inl, [839](#)
  - \_\_FFLASFFPACK\_fgemm\_bini\_INL, [839](#)
- schedule\_winograd.inl, [839](#)
  - \_\_FFLASFFPACK\_fgemm\_winograd\_INL, [840](#)
- schedule\_winograd\_acc.inl, [840](#)
  - \_\_FFLASFFPACK\_fgemm\_winograd\_acc\_INL, [841](#)
- schedule\_winograd\_acc\_ip.inl, [841](#)
  - \_\_FFLASFFPACK\_fgemm\_winograd\_acc\_ip\_INL, [841](#)
- schedule\_winograd\_ip.inl, [841](#)
  - \_\_FFLASFFPACK\_fgemm\_winograd\_ip\_INL, [842](#)
- scopy\_
  - config-blas.h, [814](#)
- sdot\_
  - config-blas.h, [813](#)
- second\_component
  - Compose< H1, H2 >, [424](#)
- Self
  - Coo< ValT, IdxT >, [431](#), [434](#)
- Self\_t
  - MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, [504](#)
  - MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, [506](#)
  - MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< Simd256\_impl< true, true, true, 2 >, 672  
Dest >, ParSeqTrait >, [508](#)
  - MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< Simd256\_impl< true, true, true, 8 >, 698  
ElementCategories::RNSElementTag >, ParSeqTrait >, [509](#)
  - MMHelper< Field, AlgoTrait, ModeCategories::DefaultTag, Simd512\_impl< true, true, true, 8 >, 726  
ParSeqTrait >, [511](#)
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, [499](#)
  - Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, [751](#)
- SELL
  - FFLAS, [76](#)
- sell.h, [909](#)
- sell\_pspmv.inl, [909](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_sell\_pspmv\_INL, [910](#)
- sell\_spmv.inl, [910](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_sell\_spmv\_INL, [911](#)
- sell\_utils.inl, [911](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_sell\_utils\_INL, [911](#)
- SELL\_ZO
  - FFLAS, [76](#)
- Sequential, [563](#)
  - numthreads, [563](#)
  - operator<<, [564](#)
  - Sequential, [563](#)
- set
  - Simd128\_impl< true, true, false, 2 >, [567](#), [569](#)
  - Simd128\_impl< true, true, false, 4 >, [577](#), [579](#)
  - Simd128\_impl< true, true, false, 8 >, [587](#), [589](#)
  - Simd128\_impl< true, true, true, 2 >, [597](#)
  - Simd128\_impl< true, true, true, 4 >, [606](#)
  - Simd128\_impl< true, true, true, 8 >, [615](#)
  - Simd256\_impl< true, false, true, 8 >, [626](#)
  - Simd256\_impl< true, true, false, 2 >, [634](#), [637](#)
  - Simd256\_impl< true, true, false, 4 >, [645](#), [648](#), [651](#)
  - Simd256\_impl< true, true, false, 8 >, [662](#), [665](#)
  - Simd256\_impl< true, true, true, 2 >, [672](#)
  - Simd256\_impl< true, true, true, 4 >, [683](#), [689](#)
  - Simd256\_impl< true, true, true, 8 >, [698](#)
  - Simd512\_impl< true, false, true, 8 >, [707](#)
  - Simd512\_impl< true, true, false, 8 >, [715](#), [718](#)
  - Simd512\_impl< true, true, true, 8 >, [726](#)
- set1
  - Simd128\_impl< true, true, false, 2 >, [567](#), [569](#)
  - Simd128\_impl< true, true, false, 4 >, [577](#), [579](#)
  - Simd128\_impl< true, true, false, 8 >, [587](#), [589](#)
  - Simd128\_impl< true, true, true, 2 >, [597](#)
  - Simd128\_impl< true, true, true, 4 >, [606](#)
  - Simd128\_impl< true, true, true, 8 >, [615](#)
  - Simd256\_impl< true, false, true, 8 >, [626](#)
  - Simd256\_impl< true, true, false, 2 >, [634](#), [636](#)
  - Simd256\_impl< true, true, false, 4 >, [645](#), [648](#), [651](#)
  - Simd256\_impl< true, true, false, 8 >, [662](#), [665](#)
  - Simd256\_impl< true, true, true, 2 >, [672](#)
  - Simd256\_impl< true, true, true, 4 >, [683](#), [689](#)
  - Simd256\_impl< true, true, true, 8 >, [698](#)
  - Simd512\_impl< true, false, true, 8 >, [707](#)
  - Simd512\_impl< true, true, false, 8 >, [715](#), [718](#)
  - Simd512\_impl< true, true, true, 8 >, [726](#)
- set\_numthreads
  - Parallel< C, P >, [522](#)
- SET\_THREADS
  - parallel.h, [1034](#)
- setErrorStream
  - Failure, [445](#)
- setNorm
  - MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, [504](#)

- MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, 506
- MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo<\_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_float\_INL, ElementCategories::RNSElementTag >, ParSeqTrait >, 510
- setOutBounds
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, 501
- sgemm\_
  - config-blas.h, 816
- sgemv\_
  - config-blas.h, 813
- sger\_
  - config-blas.h, 814
- shuffle
  - Simd128\_impl< true, true, false, 2 >, 570
  - Simd128\_impl< true, true, false, 4 >, 580
  - Simd128\_impl< true, true, false, 8 >, 590
  - Simd128\_impl< true, true, true, 2 >, 598
  - Simd128\_impl< true, true, true, 4 >, 607
  - Simd128\_impl< true, true, true, 8 >, 616
  - Simd256\_impl< true, true, false, 2 >, 638
  - Simd256\_impl< true, true, false, 4 >, 653
  - Simd256\_impl< true, true, false, 8 >, 666
  - Simd256\_impl< true, true, true, 2 >, 673
  - Simd256\_impl< true, true, true, 4 >, 684, 690
  - Simd256\_impl< true, true, true, 8 >, 699
  - Simd512\_impl< true, false, true, 8 >, 708
  - Simd512\_impl< true, true, false, 8 >, 719
  - Simd512\_impl< true, true, true, 8 >, 727
- shuffle\_twice
  - Simd256\_impl< true, true, false, 4 >, 652
  - Simd256\_impl< true, true, true, 4 >, 684, 690
- sigma
  - Sparse< \_Field, SparseMatrix\_t::SELL >, 753
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, 755
- signbits
  - Simd128\_impl< true, true, false, 8 >, 593
  - Simd128\_impl< true, true, true, 8 >, 621
  - Simd256\_impl< true, true, false, 8 >, 669
  - Simd256\_impl< true, true, true, 8 >, 704
  - Simd512\_impl< true, true, false, 8 >, 722
  - Simd512\_impl< true, true, true, 8 >, 732
- Simd
  - fflas\_simd.h, 873
- simd
  - FieldSimd< \_Field >, 447
- SIMD wrapper, 42
- simd.doxy, 873
- Simd128
  - simd128.inl, 873
- simd128.inl, 873
  - \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_INL, 873
  - Simd128, 873
- simd128\_double.inl, 874
  - \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_double\_INL, 874
  - simd128\_float.inl, 874
- Simd128\_impl< ArithType, Int, Signed, Size >, 564
- Simd128\_impl< true, false, true, 4 >, 564
- Simd128\_impl< true, false, true, 8 >, 564
- Simd128\_impl< true, true, false, 2 >, 564
  - add, 572
  - addin, 572
  - aligned\_allocator, 566
  - aligned\_vector, 566
  - alignment, 574
  - blend, 572
  - compliant, 569
  - eq, 573
  - fmadd, 572
  - fmaddin, 572
  - fmaddx, 568
  - fmaddxin, 568
  - fmsub, 573
  - fmsubin, 573
  - fmsubx, 569
  - fmsubxin, 569
  - fnmadd, 572
  - fnmaddin, 573
  - fnmaddx, 568
  - fnmaddxin, 569
  - gather, 567, 570
  - greater, 568
  - greater\_eq, 568
  - hadd\_to\_scal, 569
  - is\_same\_element, 566
  - lesser, 568
  - lesser\_eq, 568
  - load, 567, 570
  - loadu, 567, 570
  - mod, 573
  - mul, 572
  - mulhi, 568
  - mullo, 572
  - mulx, 568
  - pack, 571
  - pack\_even, 571
  - pack\_odd, 571
  - round, 573
  - scalar\_t, 566
  - set, 567, 569
  - set1, 567, 569
  - shuffle, 570
  - sll, 570
  - sll128, 573
  - sra, 567
  - srl, 570
  - srl128, 573
  - store, 567, 570
  - storeu, 567, 570

- stream, [567](#), [570](#)
- sub, [572](#)
- subin, [572](#)
- transpose, [571](#)
- type\_string, [567](#)
- unpackhi, [571](#)
- unpackhi\_intrinsic, [571](#)
- unpacklo, [571](#)
- unpacklo\_intrinsic, [570](#)
- unpacklohi, [571](#)
- valid, [569](#)
- vand, [574](#)
- vandnot, [574](#)
- vect\_size, [574](#)
- vect\_t, [566](#)
- vor, [574](#)
- vxor, [574](#)
- zero, [573](#)
- Simd128\_impl< true, true, false, 2 >::Converter, [425](#)
- t, [425](#)
- v, [425](#)
- Simd128\_impl< true, true, false, 4 >, [574](#)
- add, [581](#)
- addin, [582](#)
- aligned\_allocator, [576](#)
- aligned\_vector, [576](#)
- alignment, [584](#)
- blend, [581](#)
- compliant, [579](#)
- eq, [583](#)
- fmadd, [582](#)
- fmaddin, [582](#)
- fmaddx, [578](#)
- fmaddxin, [578](#)
- fmsub, [583](#)
- fmsubin, [583](#)
- fmsubx, [579](#)
- fmsubxin, [579](#)
- fnmadd, [582](#)
- fnmaddin, [582](#)
- fnmaddx, [578](#)
- fnmaddxin, [579](#)
- gather, [577](#), [579](#)
- greater, [578](#)
- greater\_eq, [578](#)
- hadd\_to\_scal, [579](#)
- is\_same\_element, [576](#)
- lesser, [578](#)
- lesser\_eq, [578](#)
- load, [577](#), [580](#)
- loadu, [577](#), [580](#)
- mod, [583](#)
- mul, [582](#)
- mulhi, [578](#)
- mullo, [582](#)
- mulx, [578](#)
- pack, [581](#)
- pack\_even, [581](#)
- pack\_odd, [581](#)
- round, [583](#)
- scalar\_t, [576](#)
- set, [577](#), [579](#)
- set1, [577](#), [579](#)
- shuffle, [580](#)
- sll, [580](#)
- sll128, [583](#)
- sra, [578](#)
- srl, [580](#)
- srl128, [583](#)
- store, [577](#), [580](#)
- storeu, [577](#), [580](#)
- stream, [577](#), [580](#)
- sub, [582](#)
- subin, [582](#)
- transpose, [581](#)
- type\_string, [577](#)
- unpackhi, [581](#)
- unpackhi\_intrinsic, [580](#)
- unpacklo, [581](#)
- unpacklo\_intrinsic, [580](#)
- unpacklohi, [581](#)
- valid, [579](#)
- vand, [583](#)
- vandnot, [584](#)
- vect\_size, [584](#)
- vect\_t, [577](#)
- vor, [584](#)
- vxor, [584](#)
- zero, [583](#)
- Simd128\_impl< true, true, false, 4 >::Converter, [425](#)
- t, [425](#)
- v, [425](#)
- Simd128\_impl< true, true, false, 8 >, [584](#)
- add, [592](#)
- addin, [592](#)
- aligned\_allocator, [586](#)
- aligned\_vector, [586](#)
- alignment, [594](#)
- blend, [591](#)
- compliant, [589](#)
- eq, [593](#)
- fmadd, [592](#)
- fmaddin, [592](#)
- fmaddx, [588](#)
- fmaddxin, [588](#)
- fmsub, [593](#)
- fmsubin, [593](#)
- fmsubx, [589](#)
- fmsubxin, [589](#)
- fnmadd, [592](#)
- fnmaddin, [592](#)
- fnmaddx, [588](#)
- fnmaddxin, [589](#)
- gather, [587](#), [589](#)
- get, [590](#)
- greater, [588](#)



- greater\_eq, [588](#)
- hadd\_to\_scal, [589](#)
- is\_same\_element, [586](#)
- lesser, [588](#)
- lesser\_eq, [588](#)
- load, [587](#), [590](#)
- loadu, [587](#), [590](#)
- mask\_high, [593](#)
- mod, [593](#)
- mul, [592](#)
- mulhi, [588](#)
- mulhi\_fast, [593](#)
- mullo, [588](#)
- mulx, [588](#)
- pack, [591](#)
- pack\_even, [591](#)
- pack\_odd, [591](#)
- round, [593](#)
- scalar\_t, [586](#)
- set, [587](#), [589](#)
- set1, [587](#), [589](#)
- shuffle, [590](#)
- signbits, [593](#)
- sll, [590](#)
- sll128, [594](#)
- sra, [587](#)
- srl, [590](#)
- srl128, [594](#)
- store, [587](#), [590](#)
- storeu, [587](#), [590](#)
- stream, [587](#), [590](#)
- sub, [592](#)
- subin, [592](#)
- transpose, [591](#)
- type\_string, [587](#)
- unpackhi, [591](#)
- unpackhi\_intrinsic, [591](#)
- unpacklo, [591](#)
- unpacklo\_intrinsic, [590](#)
- unpacklohi, [591](#)
- valid, [589](#)
- vand, [594](#)
- vandnot, [594](#)
- vect\_size, [594](#)
- vect\_t, [587](#)
- vor, [594](#)
- vxor, [594](#)
- zero, [593](#)
- Simd128\_impl< true, true, false, 8 >::Converter, [425](#)
- t, [426](#)
- v, [426](#)
- Simd128\_impl< true, true, true, 2 >, [594](#)
- add, [599](#)
- addin, [600](#)
- aligned\_allocator, [596](#)
- aligned\_vector, [597](#)
- alignment, [603](#)
- blend, [599](#)
- compliant, [597](#)
- eq, [602](#)
- fmadd, [600](#)
- fmaddin, [600](#)
- fmaddx, [600](#)
- fmaddxin, [601](#)
- fmsub, [601](#)
- fmsubin, [601](#)
- fmsubx, [601](#)
- fmsubxin, [602](#)
- fnmadd, [601](#)
- fnmaddin, [601](#)
- fnmaddx, [601](#)
- fnmaddxin, [601](#)
- gather, [597](#)
- greater, [602](#)
- greater\_eq, [602](#)
- hadd\_to\_scal, [602](#)
- is\_same\_element, [597](#)
- lesser, [602](#)
- lesser\_eq, [602](#)
- load, [597](#)
- loadu, [597](#)
- mod, [602](#)
- mul, [600](#)
- mulhi, [600](#)
- mullo, [600](#)
- mulx, [600](#)
- pack, [599](#)
- pack\_even, [599](#)
- pack\_odd, [599](#)
- round, [602](#)
- scalar\_t, [596](#)
- set, [597](#)
- set1, [597](#)
- shuffle, [598](#)
- sll, [598](#)
- sll128, [603](#)
- sra, [598](#)
- srl, [598](#)
- srl128, [603](#)
- store, [598](#)
- storeu, [598](#)
- stream, [598](#)
- sub, [600](#)
- subin, [600](#)
- transpose, [599](#)
- type\_string, [597](#)
- unpackhi, [599](#)
- unpackhi\_intrinsic, [598](#)
- unpacklo, [598](#)
- unpacklo\_intrinsic, [598](#)
- unpacklohi, [599](#)
- valid, [597](#)
- vand, [603](#)
- vandnot, [603](#)
- vect\_size, [603](#)
- vect\_t, [596](#)



- vor, [603](#)
- vxor, [603](#)
- zero, [603](#)
- Simd128\_impl< true, true, true, 2 >::Converter, [426](#)
- t, [426](#)
- v, [426](#)
- Simd128\_impl< true, true, true, 4 >, [604](#)
- add, [608](#)
- addin, [608](#)
- aligned\_allocator, [605](#)
- aligned\_vector, [606](#)
- alignment, [612](#)
- blend, [608](#)
- compliant, [606](#)
- eq, [611](#)
- fmadd, [609](#)
- fmaddin, [609](#)
- fmaddx, [609](#)
- fmaddxin, [609](#)
- fmsub, [610](#)
- fmsubin, [610](#)
- fmsubx, [610](#)
- fmsubxin, [610](#)
- fnmadd, [610](#)
- fnmaddin, [610](#)
- fnmaddx, [610](#)
- fnmaddxin, [610](#)
- gather, [606](#)
- greater, [611](#)
- greater\_eq, [611](#)
- hadd\_to\_scal, [611](#)
- is\_same\_element, [606](#)
- lesser, [611](#)
- lesser\_eq, [611](#)
- load, [606](#)
- loadu, [606](#)
- mod, [611](#)
- mul, [609](#)
- mulhi, [609](#)
- mullo, [609](#)
- mulx, [609](#)
- pack, [608](#)
- pack\_even, [608](#)
- pack\_odd, [608](#)
- round, [611](#)
- scalar\_t, [605](#)
- set, [606](#)
- set1, [606](#)
- shuffle, [607](#)
- sll, [607](#)
- sll128, [611](#)
- sra, [607](#)
- srl, [607](#)
- srl128, [612](#)
- store, [606](#)
- storeu, [607](#)
- stream, [607](#)
- sub, [608](#)
- subin, [609](#)
- transpose, [608](#)
- type\_string, [606](#)
- unpackhi, [607](#)
- unpackhi\_intrinsic, [607](#)
- unpacklo, [607](#)
- unpacklo\_intrinsic, [607](#)
- unpacklohi, [608](#)
- valid, [606](#)
- vand, [612](#)
- vandnot, [612](#)
- vect\_size, [612](#)
- vect\_t, [605](#)
- vor, [612](#)
- vxor, [612](#)
- zero, [611](#)
- Simd128\_impl< true, true, true, 4 >::Converter, [426](#)
- t, [426](#)
- v, [426](#)
- Simd128\_impl< true, true, true, 8 >, [612](#)
- add, [617](#)
- addin, [618](#)
- aligned\_allocator, [615](#)
- aligned\_vector, [615](#)
- alignment, [622](#)
- blend, [617](#)
- compliant, [615](#)
- eq, [620](#)
- fmadd, [618](#)
- fmaddin, [618](#)
- fmaddx, [618](#)
- fmaddxin, [619](#)
- fmsub, [619](#)
- fmsubin, [619](#)
- fmsubx, [619](#)
- fmsubxin, [620](#)
- fnmadd, [619](#)
- fnmaddin, [619](#)
- fnmaddx, [619](#)
- fnmaddxin, [619](#)
- gather, [615](#)
- get, [615](#)
- greater, [620](#)
- greater\_eq, [620](#)
- hadd\_to\_scal, [620](#)
- is\_same\_element, [615](#)
- lesser, [620](#)
- lesser\_eq, [620](#)
- load, [615](#)
- loadu, [616](#)
- mask\_high, [620](#)
- mod, [620](#)
- mul, [618](#)
- mulhi, [618](#)
- mulhi\_fast, [620](#)
- mullo, [618](#)
- mulx, [618](#)
- pack, [617](#)

- pack\_even, [617](#)
- pack\_odd, [617](#)
- round, [620](#)
- scalar\_t, [614](#)
- set, [615](#)
- set1, [615](#)
- shuffle, [616](#)
- signbits, [621](#)
- sll, [616](#)
- sll128, [621](#)
- sra, [616](#)
- srl, [616](#)
- srl128, [621](#)
- store, [616](#)
- storeu, [616](#)
- stream, [616](#)
- sub, [618](#)
- subin, [618](#)
- transpose, [617](#)
- type\_string, [615](#)
- unpackhi, [617](#)
- unpackhi\_intrinsic, [616](#)
- unpacklo, [617](#)
- unpacklo\_intrinsic, [616](#)
- unpacklohi, [617](#)
- valid, [615](#)
- vand, [621](#)
- vandnot, [621](#)
- vect\_size, [622](#)
- vect\_t, [614](#)
- vor, [621](#)
- vxor, [621](#)
- zero, [621](#)
- Simd128\_impl< true, true, true, 8 >::Converter, [426](#)
- t, [427](#)
- v, [427](#)
- simd128\_int16.inl, [874](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_int16\_INL, [875](#)
- simd128\_int32.inl, [875](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_int32\_INL, [875](#)
- simd128\_int64.inl, [875](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_int64\_INL, [876](#)
- vect\_t, [876](#)
- Simd128i\_base, [622](#)
- sll128, [622](#)
- srl128, [622](#)
- vand, [623](#)
- vandnot, [623](#)
- vect\_t, [622](#)
- vor, [623](#)
- vxor, [623](#)
- zero, [622](#)
- Simd256
- simd256.inl, [876](#)
- simd256.inl, [876](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_INL, [876](#)
- Simd256, [876](#)
- simd256\_double.inl, [876](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_double\_INL, [877](#)
- simd256\_float.inl, [877](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_float\_INL, [877](#)
- Simd256\_impl< ArithType, Int, Signed, Size >, [623](#)
- Simd256\_impl< true, false, true, 4 >, [623](#)
- Simd256\_impl< true, false, true, 8 >, [624](#)
- add, [628](#)
- addin, [628](#)
- aligned\_allocator, [625](#)
- aligned\_vector, [625](#)
- alignment, [631](#)
- blend, [628](#)
- blendv, [628](#)
- ceil, [630](#)
- compliant, [626](#)
- div, [628](#)
- eq, [629](#)
- floor, [630](#)
- fmadd, [629](#)
- fmaddin, [629](#)
- fmsub, [629](#)
- fmsubin, [629](#)
- fnmadd, [629](#)
- fnmaddin, [629](#)
- gather, [626](#)
- greater, [630](#)
- greater\_eq, [630](#)
- hadd, [630](#)
- hadd\_to\_scal, [631](#)
- is\_same\_element, [625](#)
- lesser, [629](#)
- lesser\_eq, [629](#)
- load, [626](#)
- loadu, [626](#)
- mod, [631](#)
- mul, [628](#)
- mulin, [628](#)
- pack, [627](#)
- pack\_even, [627](#)
- pack\_odd, [627](#)
- round, [630](#)
- scalar\_t, [625](#)
- set, [626](#)
- set1, [626](#)
- store, [626](#)
- storeu, [626](#)
- stream, [626](#)
- sub, [628](#)
- subin, [628](#)
- transpose, [627](#)
- type\_string, [625](#)
- unpackhi, [627](#)

- unpackhi\_intrinsic, [627](#)
- unpacklo, [627](#)
- unpacklo\_intrinsic, [627](#)
- unpacklohi, [627](#)
- valid, [625](#)
- vand, [630](#)
- vandnot, [630](#)
- vect\_size, [631](#)
- vect\_t, [625](#)
- vor, [630](#)
- vxor, [630](#)
- zero, [626](#)
- Simd256\_impl< true, false, true, 8 >::Converter, [427](#)
- t, [427](#)
- v, [427](#)
- Simd256\_impl< true, true, false, 2 >, [631](#)
- add, [639](#)
- addin, [639](#)
- aligned\_allocator, [633](#)
- aligned\_vector, [633](#)
- alignment, [641](#)
- blend, [639](#)
- compliant, [636](#)
- eq, [640](#)
- fmadd, [640](#)
- fmaddin, [640](#)
- fmaddx, [635](#)
- fmaddxin, [636](#)
- fmsub, [640](#)
- fmubin, [640](#)
- fmubx, [636](#)
- fmubxin, [636](#)
- fnmadd, [640](#)
- fnmaddin, [640](#)
- fnmaddx, [636](#)
- fnmaddxin, [636](#)
- gather, [634](#), [637](#)
- greater, [635](#)
- greater\_eq, [635](#)
- hadd\_to\_scal, [636](#)
- half\_t, [634](#)
- is\_same\_element, [633](#)
- lesser, [635](#)
- lesser\_eq, [635](#)
- load, [634](#), [637](#)
- loadu, [634](#), [637](#)
- mod, [641](#)
- mul, [640](#)
- mulhi, [635](#)
- mullo, [640](#)
- mulx, [635](#)
- pack, [638](#)
- pack\_even, [638](#)
- pack\_odd, [638](#)
- round, [641](#)
- scalar\_t, [633](#)
- set, [634](#), [637](#)
- set1, [634](#), [636](#)
- shuffle, [638](#)
- simdHalf, [633](#)
- sll, [637](#)
- sra, [635](#)
- srl, [638](#)
- store, [634](#), [637](#)
- storeu, [635](#), [637](#)
- stream, [635](#), [637](#)
- sub, [639](#)
- subin, [639](#)
- transpose, [639](#)
- type\_string, [634](#)
- unpackhi, [638](#)
- unpackhi\_intrinsic, [638](#)
- unpacklo, [638](#)
- unpacklo\_intrinsic, [638](#)
- unpacklohi, [638](#)
- valid, [636](#)
- vect\_size, [641](#)
- vect\_t, [634](#)
- zero, [641](#)
- Simd256\_impl< true, true, false, 2 >::Converter, [427](#)
- t, [427](#)
- v, [427](#)
- Simd256\_impl< true, true, false, 4 >, [641](#)
- add, [655](#)
- addin, [655](#), [656](#)
- aligned\_allocator, [645](#)
- aligned\_vector, [645](#)
- alignment, [659](#)
- blend, [655](#)
- compliant, [651](#)
- eq, [658](#)
- fmadd, [656](#), [657](#)
- fmaddin, [657](#)
- fmaddx, [647](#), [649](#)
- fmaddxin, [647](#), [650](#)
- fmsub, [657](#), [658](#)
- fmubin, [658](#)
- fmubx, [647](#), [650](#)
- fmubxin, [648](#), [650](#)
- fnmadd, [657](#)
- fnmaddin, [657](#)
- fnmaddx, [647](#), [650](#)
- fnmaddxin, [647](#), [650](#)
- gather, [646](#), [648](#), [651](#)
- greater, [646](#), [649](#)
- greater\_eq, [647](#), [649](#)
- hadd\_to\_scal, [648](#), [650](#)
- half\_t, [645](#)
- is\_same\_element, [645](#), [650](#)
- lesser, [646](#), [649](#)
- lesser\_eq, [647](#), [649](#)
- load, [646](#), [648](#), [652](#)
- loadu, [646](#), [648](#), [652](#)
- mod, [658](#)
- mul, [656](#)
- mulhi, [647](#), [649](#)

- mullo, [656](#)
- mulx, [647](#), [649](#)
- pack, [654](#)
- pack\_even, [654](#)
- pack\_odd, [654](#)
- round, [658](#)
- scalar\_t, [645](#)
- set, [645](#), [648](#), [651](#)
- set1, [645](#), [648](#), [651](#)
- shuffle, [653](#)
- shuffle\_twice, [652](#)
- simdHalf, [645](#)
- sll, [652](#)
- sra, [646](#), [649](#)
- srl, [652](#)
- store, [646](#), [648](#), [652](#)
- storeu, [646](#), [649](#), [652](#)
- stream, [646](#), [649](#), [652](#)
- sub, [656](#)
- subin, [656](#)
- transpose, [655](#)
- type\_string, [645](#), [648](#)
- unpackhi, [653](#)
- unpackhi\_intrinsic, [653](#)
- unpacklo, [653](#)
- unpacklo\_intrinsic, [653](#)
- unpacklohi, [654](#)
- valid, [650](#)
- vand, [659](#)
- vandnot, [659](#)
- vect\_size, [659](#)
- vect\_t, [645](#)
- vor, [659](#)
- vxor, [659](#)
- zero, [659](#)
- Simd256\_impl< true, true, false, 4 >::Converter, [427](#)
- t, [428](#)
- v, [428](#)
- Simd256\_impl< true, true, false, 8 >, [660](#)
- add, [667](#)
- addin, [667](#)
- aligned\_allocator, [662](#)
- aligned\_vector, [662](#)
- alignment, [669](#)
- blend, [667](#)
- compliant, [665](#)
- eq, [668](#)
- fmadd, [668](#)
- fmaddin, [668](#)
- fmaddx, [664](#)
- fmaddxin, [664](#)
- fmsub, [668](#)
- fmsubin, [668](#)
- fmsubx, [664](#)
- fmsubxin, [664](#)
- fnmadd, [668](#)
- fnmaddin, [668](#)
- fnmaddx, [664](#)
- fnmaddxin, [664](#)
- gather, [662](#), [665](#)
- get, [665](#)
- greater, [663](#)
- greater\_eq, [663](#)
- hadd\_to\_scal, [664](#)
- half\_t, [662](#)
- is\_same\_element, [662](#)
- lesser, [663](#)
- lesser\_eq, [663](#)
- load, [662](#), [665](#)
- loadu, [663](#), [665](#)
- mask\_high, [668](#)
- mod, [669](#)
- mul, [667](#)
- mulhi, [664](#)
- mulhi\_fast, [669](#)
- mullo, [663](#)
- mulx, [664](#)
- pack, [667](#)
- pack\_even, [666](#)
- pack\_odd, [667](#)
- round, [668](#)
- scalar\_t, [662](#)
- set, [662](#), [665](#)
- set1, [662](#), [665](#)
- shuffle, [666](#)
- signbits, [669](#)
- simdHalf, [662](#)
- sll, [666](#)
- sra, [663](#)
- srl, [666](#)
- store, [663](#), [665](#)
- storeu, [663](#), [665](#)
- stream, [663](#), [666](#)
- sub, [667](#)
- subin, [667](#)
- transpose, [667](#)
- type\_string, [662](#)
- unpackhi, [666](#)
- unpackhi\_intrinsic, [666](#)
- unpacklo, [666](#)
- unpacklo\_intrinsic, [666](#)
- unpacklohi, [666](#)
- valid, [665](#)
- vect\_size, [669](#)
- vect\_t, [662](#)
- zero, [669](#)
- Simd256\_impl< true, true, false, 8 >::Converter, [428](#)
- t, [428](#)
- v, [428](#)
- Simd256\_impl< true, true, true, 2 >, [669](#)
- add, [675](#)
- addin, [675](#)
- aligned\_allocator, [671](#)
- aligned\_vector, [671](#)
- alignment, [678](#)
- blend, [675](#)

- compliant, [672](#)
- eq, [677](#)
- fmadd, [676](#)
- fmaddin, [676](#)
- fmaddx, [676](#)
- fmaddxin, [676](#)
- fmsub, [677](#)
- fmsubin, [677](#)
- fmsubx, [677](#)
- fmsubxin, [677](#)
- fnmadd, [676](#)
- fnmaddin, [676](#)
- fnmaddx, [676](#)
- fnmaddxin, [676](#)
- gather, [672](#)
- greater, [677](#)
- greater\_eq, [677](#)
- hadd\_to\_scal, [678](#)
- half\_t, [671](#)
- is\_same\_element, [672](#)
- lesser, [677](#)
- lesser\_eq, [677](#)
- load, [672](#)
- loadu, [673](#)
- mod, [678](#)
- mul, [675](#)
- mulhi, [675](#)
- mullo, [675](#)
- mulx, [676](#)
- pack, [674](#)
- pack\_even, [674](#)
- pack\_odd, [674](#)
- round, [678](#)
- scalar\_t, [671](#)
- set, [672](#)
- set1, [672](#)
- shuffle, [673](#)
- simdHalf, [671](#)
- sll, [673](#)
- sra, [673](#)
- srl, [673](#)
- store, [673](#)
- storeu, [673](#)
- stream, [673](#)
- sub, [675](#)
- subin, [675](#)
- transpose, [674](#)
- type\_string, [672](#)
- unpackhi, [674](#)
- unpackhi\_intrinsic, [673](#)
- unpacklo, [674](#)
- unpacklo\_intrinsic, [673](#)
- unpacklohi, [674](#)
- valid, [672](#)
- vect\_size, [678](#)
- vect\_t, [671](#)
- zero, [678](#)
- Simd256\_impl< true, true, true, 2 >::Converter, [428](#)
- t, [428](#)
- v, [428](#)
- Simd256\_impl< true, true, true, 4 >, [678](#)
  - add, [685](#), [691](#)
  - addin, [685](#), [691](#)
  - aligned\_allocator, [682](#)
  - aligned\_vector, [682](#)
  - alignment, [695](#)
  - blend, [685](#), [691](#)
  - compliant, [682](#), [688](#)
  - eq, [687](#), [694](#)
  - fmadd, [686](#), [692](#)
  - fmaddin, [686](#), [692](#)
  - fmaddx, [686](#), [692](#)
  - fmaddxin, [686](#), [693](#)
  - fmsub, [687](#), [693](#)
  - fmsubin, [687](#), [693](#)
  - fmsubx, [687](#), [693](#)
  - fmsubxin, [687](#), [694](#)
  - fnmadd, [686](#), [693](#)
  - fnmaddin, [687](#), [693](#)
  - fnmaddx, [687](#), [693](#)
  - fnmaddxin, [687](#), [693](#)
  - gather, [683](#), [689](#)
  - greater, [688](#), [694](#)
  - greater\_eq, [688](#), [694](#)
  - hadd\_to\_scal, [688](#), [694](#)
  - half\_t, [682](#)
  - is\_same\_element, [682](#), [695](#)
  - lesser, [688](#), [694](#)
  - lesser\_eq, [688](#), [694](#)
  - load, [683](#), [689](#)
  - loadu, [683](#), [689](#)
  - mod, [688](#), [694](#)
  - mul, [686](#), [692](#)
  - mulhi, [686](#), [692](#)
  - mullo, [686](#), [692](#)
  - mulx, [686](#), [692](#)
  - pack, [685](#), [691](#)
  - pack\_even, [684](#), [691](#)
  - pack\_odd, [685](#), [691](#)
  - round, [688](#), [694](#)
  - scalar\_t, [682](#)
  - set, [683](#), [689](#)
  - set1, [683](#), [689](#)
  - shuffle, [684](#), [690](#)
  - shuffle\_twice, [684](#), [690](#)
  - simdHalf, [682](#)
  - sll, [683](#), [690](#)
  - sra, [684](#), [690](#)
  - srl, [684](#), [690](#)
  - store, [683](#), [689](#)
  - storeu, [683](#), [689](#)
  - stream, [683](#), [689](#)
  - sub, [685](#), [692](#)
  - subin, [685](#), [692](#)
  - transpose, [685](#), [691](#)
  - type\_string, [682](#), [688](#)

- unpackhi, [684](#), [690](#)
- unpackhi\_intrinsic, [684](#), [690](#)
- unpacklo, [684](#), [690](#)
- unpacklo\_intrinsic, [684](#), [690](#)
- unpacklohi, [684](#), [690](#)
- valid, [682](#), [688](#)
- vand, [695](#)
- vandnot, [695](#)
- vect\_size, [695](#)
- vect\_t, [682](#)
- vor, [695](#)
- vxor, [695](#)
- zero, [695](#)
- Simd256\_impl< true, true, true, 4 >::Converter, [428](#)
- t, [429](#)
- v, [429](#)
- Simd256\_impl< true, true, true, 8 >, [695](#)
- add, [701](#)
- addin, [701](#)
- aligned\_allocator, [698](#)
- aligned\_vector, [698](#)
- alignment, [704](#)
- blend, [700](#)
- compliant, [698](#)
- eq, [703](#)
- fmadd, [701](#)
- fmaddin, [701](#)
- fmaddx, [702](#)
- fmaddxin, [702](#)
- fmsub, [702](#)
- fmsubin, [702](#)
- fmsubx, [703](#)
- fmsubxin, [703](#)
- fnmadd, [702](#)
- fnmaddin, [702](#)
- fnmaddx, [702](#)
- fnmaddxin, [702](#)
- gather, [698](#)
- get, [698](#)
- greater, [703](#)
- greater\_eq, [703](#)
- hadd\_to\_scal, [703](#)
- half\_t, [697](#)
- is\_same\_element, [698](#)
- lesser, [703](#)
- lesser\_eq, [703](#)
- load, [699](#)
- loadu, [699](#)
- mask\_high, [703](#)
- mod, [704](#)
- mul, [701](#)
- mulhi, [701](#)
- mulhi\_fast, [703](#)
- mullo, [701](#)
- mulx, [701](#)
- pack, [700](#)
- pack\_even, [700](#)
- pack\_odd, [700](#)
- round, [703](#)
- scalar\_t, [697](#)
- set, [698](#)
- set1, [698](#)
- shuffle, [699](#)
- signbits, [704](#)
- simdHalf, [698](#)
- sll, [699](#)
- sra, [699](#)
- srl, [699](#)
- store, [699](#)
- storeu, [699](#)
- stream, [699](#)
- sub, [701](#)
- subin, [701](#)
- transpose, [700](#)
- type\_string, [698](#)
- unpackhi, [700](#)
- unpackhi\_intrinsic, [700](#)
- unpacklo, [700](#)
- unpacklo\_intrinsic, [699](#)
- unpacklohi, [700](#)
- valid, [698](#)
- vect\_size, [704](#)
- vect\_t, [697](#)
- zero, [704](#)
- Simd256\_impl< true, true, true, 8 >::Converter, [429](#)
- t, [429](#)
- v, [429](#)
- simd256\_int16.inl, [877](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_int16\_INL, [878](#)
- simd256\_int32.inl, [878](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_int32\_INL, [878](#)
- simd256\_int64.inl, [878](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_int64\_INL, [878](#)
- vect\_t, [878](#)
- Simd256fp\_base, [704](#)
- Simd256i\_base, [705](#)
- vect\_t, [705](#)
- zero, [705](#)
- Simd512
- simd512.inl, [879](#)
- simd512.inl, [879](#)
- \_\_FFLASFFPACK\_simd512\_INL, [879](#)
- Simd512, [879](#)
- simd512\_double.inl, [879](#)
- \_\_FFLASFFPACK\_simd512\_double\_INL, [880](#)
- simd512\_float.inl, [880](#)
- \_\_FFLASFFPACK\_simd512\_float\_INL, [880](#)
- Simd512\_impl< ArithType, Int, Signed, Size >, [705](#)
- Simd512\_impl< true, false, true, 4 >, [705](#)
- Simd512\_impl< true, false, true, 8 >, [705](#)
- add, [710](#)
- addin, [710](#)
- aligned\_allocator, [707](#)

- aligned\_vector, 707
- alignment, 712
- blend, 709
- blendv, 710
- ceil, 712
- compliant, 707
- div, 710
- eq, 711
- floor, 712
- fmadd, 710
- fmaddin, 711
- fmsub, 711
- fmsubin, 711
- fnmadd, 711
- fnmaddin, 711
- gather, 708
- greater, 711
- greater\_eq, 712
- hadd, 712
- hadd\_to\_scal, 712
- is\_same\_element, 707
- lesser, 711
- lesser\_eq, 711
- load, 708
- loadu, 708
- mul, 710
- mulin, 710
- pack, 709
- pack\_even, 709
- pack\_odd, 709
- round, 712
- scalar\_t, 707
- set, 707
- set1, 707
- shuffle, 708
- store, 708
- storeu, 708
- stream, 708
- sub, 710
- subin, 710
- transpose, 709
- type\_string, 707
- unpackhi, 709
- unpackhi\_intrinsic, 708
- unpacklo, 709
- unpacklo\_intrinsic, 708
- unpacklohi, 709
- valid, 707
- vect\_size, 712
- vect\_t, 707
- zero, 707
- Simd512\_impl< true, true, false, 8 >, 712
  - add, 721
  - addin, 721
  - aligned\_allocator, 715
  - aligned\_vector, 715
  - alignment, 723
  - blend, 721
  - compliant, 718
  - eq, 722
  - fmadd, 721
  - fmaddin, 721
  - fmaddx, 717
  - fmaddxin, 717
  - fmsub, 722
  - fmsubin, 722
  - fmsubx, 717
  - fmsubxin, 718
  - fnmadd, 721
  - fnmaddin, 722
  - fnmaddx, 717
  - fnmaddxin, 717
  - gather, 715, 718
  - greater, 716
  - greater\_eq, 716
  - hadd\_to\_scal, 718
  - half\_t, 715
  - is\_same\_element, 715
  - lesser, 716
  - lesser\_eq, 717
  - load, 716, 719
  - loadu, 716, 719
  - mask\_high, 722
  - maskstore, 716, 719
  - mod, 722
  - mul, 721
  - mulhi, 717
  - mulhi\_fast, 722
  - mullo, 717
  - mulx, 717
  - pack, 720
  - pack\_even, 720
  - pack\_odd, 720
  - round, 722
  - scalar\_t, 715
  - set, 715, 718
  - set1, 715, 718
  - shuffle, 719
  - signbits, 722
  - simdHalf, 715
  - sll, 719
  - sra, 716
  - srl, 719
  - store, 716, 719
  - storeu, 716, 719
  - stream, 716, 719
  - sub, 721
  - subin, 721
  - transpose, 720
  - type\_string, 715
  - unpackhi, 720
  - unpackhi\_intrinsic, 720
  - unpacklo, 720
  - unpacklo\_intrinsic, 719
  - unpacklohi, 720
  - valid, 718

- vand, [723](#)
- vandnot, [723](#)
- vect\_size, [723](#)
- vect\_t, [715](#)
- vor, [723](#)
- vxor, [723](#)
- zero, [723](#)
- Simd512\_impl< true, true, false, 8 >::Converter, [429](#)
- t, [429](#)
- v, [429](#)
- Simd512\_impl< true, true, true, 8 >, [723](#)
- add, [729](#)
- addin, [729](#)
- aligned\_allocator, [726](#)
- aligned\_vector, [726](#)
- alignment, [733](#)
- blend, [729](#)
- compliant, [726](#)
- eq, [731](#)
- fmadd, [730](#)
- fmaddin, [730](#)
- fmaddx, [730](#)
- fmaddxin, [730](#)
- fmsub, [731](#)
- fmsubin, [731](#)
- fmsubx, [731](#)
- fmsubxin, [731](#)
- fnmadd, [730](#)
- fnmaddin, [730](#)
- fnmaddx, [730](#)
- fnmaddxin, [730](#)
- gather, [726](#)
- greater, [731](#)
- greater\_eq, [731](#)
- hadd\_to\_scal, [731](#)
- half\_t, [725](#)
- is\_same\_element, [726](#)
- lesser, [731](#)
- lesser\_eq, [731](#)
- load, [727](#)
- loadu, [727](#)
- mask\_high, [732](#)
- maskstore, [727](#)
- mod, [732](#)
- mul, [729](#)
- mulhi, [729](#)
- mulhi\_fast, [732](#)
- mullo, [729](#)
- mulx, [729](#)
- pack, [728](#)
- pack\_even, [728](#)
- pack\_odd, [728](#)
- round, [732](#)
- scalar\_t, [725](#)
- set, [726](#)
- set1, [726](#)
- shuffle, [727](#)
- signbits, [732](#)
- simdHalf, [725](#)
- sll, [727](#)
- sra, [727](#)
- srl, [727](#)
- store, [727](#)
- storeu, [727](#)
- stream, [727](#)
- sub, [729](#)
- subin, [729](#)
- transpose, [728](#)
- type\_string, [726](#)
- unpackhi, [728](#)
- unpackhi\_intrinsic, [728](#)
- unpacklo, [728](#)
- unpacklo\_intrinsic, [728](#)
- unpacklohi, [728](#)
- valid, [726](#)
- vand, [732](#)
- vandnot, [732](#)
- vect\_size, [733](#)
- vect\_t, [725](#)
- vor, [732](#)
- vxor, [732](#)
- zero, [732](#)
- Simd512\_impl< true, true, true, 8 >::Converter, [429](#)
- t, [430](#)
- v, [430](#)
- simd512\_int32.inl, [880](#)
- \_\_FFLASFFPACK\_simd512\_int32\_INL, [880](#)
- simd512\_int64.inl, [881](#)
- \_simd512\_int64\_INL, [881](#)
- vect\_t, [881](#)
- Simd512i\_base, [733](#)
- vand, [734](#)
- vandnot, [734](#)
- vect\_t, [733](#)
- vor, [733](#)
- vxor, [733](#)
- zero, [733](#)
- SIMD\_INT
- fflas\_simd.h, [872](#)
- simd\_modular.inl, [881](#)
- SimdChooser< T, bool, bool >, [734](#)
- SimdChooser< T, false, b >, [734](#)
- value, [734](#)
- SimdChooser< T, true, false >, [734](#)
- value, [735](#)
- SimdChooser< T, true, true >, [735](#)
- value, [735](#)
- simdHalf
- Simd256\_impl< true, true, false, 2 >, [633](#)
- Simd256\_impl< true, true, false, 4 >, [645](#)
- Simd256\_impl< true, true, false, 8 >, [662](#)
- Simd256\_impl< true, true, true, 2 >, [671](#)
- Simd256\_impl< true, true, true, 4 >, [682](#)
- Simd256\_impl< true, true, true, 8 >, [698](#)
- Simd512\_impl< true, true, false, 8 >, [715](#)
- Simd512\_impl< true, true, true, 8 >, [725](#)



SimdSparseMatrix  
     FFLAS, 72  
 simdToType< T >, 735  
 Single, 735  
 size  
     BlockTransposeSIMD< Field, Simd, >, 409  
     Info, 479, 480  
     RNSInteger< RNS >, 543  
     RNSIntegerMod< RNS >, 547  
 SIZEOF\_\_\_INT64\_T  
     config.h, 820  
 SIZEOF\_CHAR  
     config.h, 819  
 SIZEOF\_INT  
     config.h, 819  
 SIZEOF\_LONG  
     config.h, 819  
 SIZEOF\_LONG\_LONG  
     config.h, 820  
 SIZEOF\_SHORT  
     config.h, 820  
 sll  
     ScalFunctionsBase< Element, typename enable\_if<  
         is\_integral< Element >::value >::type >, 562  
     Simd128\_impl< true, true, false, 2 >, 570  
     Simd128\_impl< true, true, false, 4 >, 580  
     Simd128\_impl< true, true, false, 8 >, 590  
     Simd128\_impl< true, true, true, 2 >, 598  
     Simd128\_impl< true, true, true, 4 >, 607  
     Simd128\_impl< true, true, true, 8 >, 616  
     Simd256\_impl< true, true, false, 2 >, 637  
     Simd256\_impl< true, true, false, 4 >, 652  
     Simd256\_impl< true, true, false, 8 >, 666  
     Simd256\_impl< true, true, true, 2 >, 673  
     Simd256\_impl< true, true, true, 4 >, 683, 690  
     Simd256\_impl< true, true, true, 8 >, 699  
     Simd512\_impl< true, true, false, 8 >, 719  
     Simd512\_impl< true, true, true, 8 >, 727  
 sll128  
     Simd128\_impl< true, true, false, 2 >, 573  
     Simd128\_impl< true, true, false, 4 >, 583  
     Simd128\_impl< true, true, false, 8 >, 594  
     Simd128\_impl< true, true, true, 2 >, 603  
     Simd128\_impl< true, true, true, 4 >, 611  
     Simd128\_impl< true, true, true, 8 >, 621  
     Simd128i\_base, 622  
 Solve  
     FFPACK, 328, 372  
 solve.C, 805  
     main, 805  
 Solve\_modular\_double  
     ffpack.C, 999  
     ffpack\_c.h, 1019  
 solveLB  
     FFPACK, 348, 372  
 solveLB2  
     FFPACK, 348, 372  
 solveLB2\_modular\_double  
     ffpack.C, 1000  
     ffpack\_c.h, 1020  
 solveLB\_modular\_double  
     ffpack.C, 1000  
     ffpack\_c.h, 1019  
 Sparse< \_Field, SparseMatrix\_t::COO >, 735  
     col, 736  
     dat, 736  
     delayed, 736  
     Field, 736  
     kmax, 736  
     m, 736  
     maxrow, 737  
     n, 736  
     nElements, 737  
     nnz, 736  
     row, 736  
 Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, 737  
     col, 738  
     cst, 738  
     dat, 738  
     delayed, 738  
     Field, 737  
     kmax, 738  
     m, 738  
     maxrow, 738  
     n, 738  
     nElements, 738  
     nnz, 738  
     row, 738  
 Sparse< \_Field, SparseMatrix\_t::CSR >, 739  
     col, 740  
     dat, 740  
     delayed, 739  
     Field, 739  
     kmax, 739  
     m, 739  
     maxrow, 740  
     n, 739  
     nElements, 740  
     nnz, 739  
     st, 740  
     stend, 740  
 Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, 740  
     col, 741  
     dat, 741  
     delayed, 741  
     Field, 741  
     kmax, 741  
     m, 741  
     maxrow, 741  
     n, 741  
     nElements, 741  
     nMOnes, 742  
     nnz, 741  
     nOnes, 742  
     nOthers, 742  
     st, 741

- Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, 742
  - col, 743
  - cst, 743
  - dat, 744
  - delayed, 743
  - Field, 743
  - kmax, 743
  - m, 743
  - maxrow, 743
  - n, 743
  - nElements, 743
  - nnz, 743
  - st, 743
  - stend, 743
- Sparse< \_Field, SparseMatrix\_t::ELL >, 744
  - col, 745
  - dat, 745
  - delayed, 744
  - Field, 744
  - kmax, 744
  - ld, 745
  - m, 745
  - maxrow, 745
  - n, 745
  - nElements, 745
  - nnz, 745
- Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, 745
  - chunk, 746
  - col, 747
  - dat, 747
  - delayed, 746
  - kmax, 746
  - ld, 746
  - m, 746
  - maxrow, 746
  - n, 746
  - nChunks, 747
  - nElements, 746
  - nnz, 746
- Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, 747
  - chunk, 748
  - col, 748
  - cst, 747
  - dat, 748
  - delayed, 747
  - kmax, 748
  - ld, 748
  - m, 748
  - maxrow, 748
  - n, 748
  - nChunks, 748
  - nElements, 748
  - nnz, 748
- Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, 749
  - col, 750
  - cst, 749
  - dat, 750
  - delayed, 749
  - Field, 749
  - kmax, 749
  - ld, 750
  - m, 749
  - maxrow, 750
  - n, 749
  - nElements, 750
  - nnz, 750
- Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, 750
  - dat, 751
  - delayed, 751
  - Field, 751
  - kmax, 751
  - m, 751
  - maxrow, 751
  - mone, 751
  - n, 751
  - nElements, 751
  - nnz, 751
  - one, 751
  - Self\_t, 751
- Sparse< \_Field, SparseMatrix\_t::SELL >, 752
  - chunk, 752
  - chunkSize, 753
  - col, 753
  - dat, 754
  - delayed, 752
  - Field, 752
  - kmax, 753
  - m, 753
  - maxrow, 753
  - n, 753
  - nChunks, 753
  - nElements, 753
  - nnz, 753
  - perm, 753
  - sigma, 753
  - st, 753
- Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, 754
  - chunk, 755
  - chunkSize, 756
  - col, 756
  - cst, 755
  - dat, 756
  - delayed, 755
  - Field, 754
  - kmax, 755
  - m, 755
  - maxrow, 755
  - n, 755
  - nChunks, 755
  - nElements, 755
  - nnz, 755
  - perm, 755
  - sigma, 755
  - st, 755
- Sparse< Field, SparseMatrix\_t, IdxT, PtrT >, 735
  - sparse\_delete

- FFLAS, [146–150](#), [153](#)
- sparse\_init
  - FFLAS, [145–150](#), [153](#)
- sparse\_matrix\_traits.h, [912](#)
- sparse\_print
  - FFLAS, [147](#), [150](#), [153](#)
- SparseMatrix\_t
  - FFLAS, [76](#)
- SpecRankProfile
  - FFPACK, [371](#)
- SpecRankProfile\_modular\_double
  - ffpack.C, [999](#)
  - ffpack\_c.h, [1018](#)
- splitt
  - parallel.h, [1039](#)
- SPLITTER
  - parallel.h, [1039](#)
- splitting\_0
  - parallel.h, [1038](#)
- splitting\_1
  - parallel.h, [1038](#)
- splitting\_2
  - parallel.h, [1038](#)
- splitting\_3
  - parallel.h, [1038](#)
- SpMat< Field, flag >, [756](#)
  - \_coo, [756](#)
  - \_csr, [756](#)
  - \_ell, [756](#)
- square\_inplace
  - FFLAS::\_ftranspose\_impl, [190](#)
- sra
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, [562](#)
  - Simd128\_impl< true, true, false, 2 >, [567](#)
  - Simd128\_impl< true, true, false, 4 >, [578](#)
  - Simd128\_impl< true, true, false, 8 >, [587](#)
  - Simd128\_impl< true, true, true, 2 >, [598](#)
  - Simd128\_impl< true, true, true, 4 >, [607](#)
  - Simd128\_impl< true, true, true, 8 >, [616](#)
  - Simd256\_impl< true, true, false, 2 >, [635](#)
  - Simd256\_impl< true, true, false, 4 >, [646](#), [649](#)
  - Simd256\_impl< true, true, false, 8 >, [663](#)
  - Simd256\_impl< true, true, true, 2 >, [673](#)
  - Simd256\_impl< true, true, true, 4 >, [684](#), [690](#)
  - Simd256\_impl< true, true, true, 8 >, [699](#)
  - Simd512\_impl< true, true, false, 8 >, [716](#)
  - Simd512\_impl< true, true, true, 8 >, [727](#)
- srl
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, [562](#)
  - Simd128\_impl< true, true, false, 2 >, [570](#)
  - Simd128\_impl< true, true, false, 4 >, [580](#)
  - Simd128\_impl< true, true, false, 8 >, [590](#)
  - Simd128\_impl< true, true, true, 2 >, [598](#)
  - Simd128\_impl< true, true, true, 4 >, [607](#)
  - Simd128\_impl< true, true, true, 8 >, [616](#)
  - Simd256\_impl< true, true, false, 2 >, [638](#)
- Simd256\_impl< true, true, false, 4 >, [652](#)
- Simd256\_impl< true, true, false, 8 >, [666](#)
- Simd256\_impl< true, true, true, 2 >, [673](#)
- Simd256\_impl< true, true, true, 4 >, [684](#), [690](#)
- Simd256\_impl< true, true, true, 8 >, [699](#)
- Simd512\_impl< true, true, false, 8 >, [719](#)
- Simd512\_impl< true, true, true, 8 >, [727](#)
- srl128
  - Simd128\_impl< true, true, false, 2 >, [573](#)
  - Simd128\_impl< true, true, false, 4 >, [583](#)
  - Simd128\_impl< true, true, false, 8 >, [594](#)
  - Simd128\_impl< true, true, true, 2 >, [603](#)
  - Simd128\_impl< true, true, true, 4 >, [612](#)
  - Simd128\_impl< true, true, true, 8 >, [621](#)
  - Simd128i\_base, [622](#)
- sscal\_
  - config-blas.h, [815](#)
- ST
  - fflas\_transpose.h, [914](#)
- st
  - Sparse< \_Field, SparseMatrix\_t::CSR >, [740](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, [741](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, [743](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL >, [753](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, [755](#)
- StatsMatrix, [756](#)
  - averageCol, [758](#)
  - averageColDifference, [758](#)
  - averageRow, [758](#)
  - averageRowDifference, [758](#)
  - coldim, [757](#)
  - denseCols, [759](#)
  - denseRows, [759](#)
  - deviationCol, [758](#)
  - deviationColDifference, [758](#)
  - deviationRow, [758](#)
  - deviationRowDifference, [758](#)
  - maxCol, [758](#)
  - maxColDifference, [758](#)
  - maxRow, [757](#)
  - maxRowDifference, [758](#)
  - minCol, [758](#)
  - minColDifference, [758](#)
  - minRow, [758](#)
  - minRowDifference, [758](#)
  - nDenseCols, [759](#)
  - nDenseRows, [759](#)
  - nEmptyCols, [759](#)
  - nEmptyColsEnd, [759](#)
  - nEmptyRows, [759](#)
  - nMOnes, [757](#)
  - nnz, [757](#)
  - nOnes, [757](#)
  - nOthers, [757](#)
  - rowdim, [757](#)
- STD\_RECINT\_SIZE
  - benchmark-fgemm-mp.C, [787](#)

- benchmark-fgemv-mp.C, [790](#)
- STDC\_HEADERS
  - config.h, [820](#)
- stend
  - Sparse< \_Field, SparseMatrix\_t::CSR >, [740](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, [743](#)
- store
  - Simd128\_impl< true, true, false, 2 >, [567](#), [570](#)
  - Simd128\_impl< true, true, false, 4 >, [577](#), [580](#)
  - Simd128\_impl< true, true, false, 8 >, [587](#), [590](#)
  - Simd128\_impl< true, true, true, 2 >, [598](#)
  - Simd128\_impl< true, true, true, 4 >, [606](#)
  - Simd128\_impl< true, true, true, 8 >, [616](#)
  - Simd256\_impl< true, false, true, 8 >, [626](#)
  - Simd256\_impl< true, true, false, 2 >, [634](#), [637](#)
  - Simd256\_impl< true, true, false, 4 >, [646](#), [648](#), [652](#)
  - Simd256\_impl< true, true, false, 8 >, [663](#), [665](#)
  - Simd256\_impl< true, true, true, 2 >, [673](#)
  - Simd256\_impl< true, true, true, 4 >, [683](#), [689](#)
  - Simd256\_impl< true, true, true, 8 >, [699](#)
  - Simd512\_impl< true, false, true, 8 >, [708](#)
  - Simd512\_impl< true, true, false, 8 >, [716](#), [719](#)
  - Simd512\_impl< true, true, true, 8 >, [727](#)
- storeu
  - Simd128\_impl< true, true, false, 2 >, [567](#), [570](#)
  - Simd128\_impl< true, true, false, 4 >, [577](#), [580](#)
  - Simd128\_impl< true, true, false, 8 >, [587](#), [590](#)
  - Simd128\_impl< true, true, true, 2 >, [598](#)
  - Simd128\_impl< true, true, true, 4 >, [607](#)
  - Simd128\_impl< true, true, true, 8 >, [616](#)
  - Simd256\_impl< true, false, true, 8 >, [626](#)
  - Simd256\_impl< true, true, false, 2 >, [635](#), [637](#)
  - Simd256\_impl< true, true, false, 4 >, [646](#), [649](#), [652](#)
  - Simd256\_impl< true, true, false, 8 >, [663](#), [665](#)
  - Simd256\_impl< true, true, true, 2 >, [673](#)
  - Simd256\_impl< true, true, true, 4 >, [683](#), [689](#)
  - Simd256\_impl< true, true, true, 8 >, [699](#)
  - Simd512\_impl< true, false, true, 8 >, [708](#)
  - Simd512\_impl< true, true, false, 8 >, [716](#), [719](#)
  - Simd512\_impl< true, true, true, 8 >, [727](#)
- stream
  - Simd128\_impl< true, true, false, 2 >, [567](#), [570](#)
  - Simd128\_impl< true, true, false, 4 >, [577](#), [580](#)
  - Simd128\_impl< true, true, false, 8 >, [587](#), [590](#)
  - Simd128\_impl< true, true, true, 2 >, [598](#)
  - Simd128\_impl< true, true, true, 4 >, [607](#)
  - Simd128\_impl< true, true, true, 8 >, [616](#)
  - Simd256\_impl< true, false, true, 8 >, [626](#)
  - Simd256\_impl< true, true, false, 2 >, [635](#), [637](#)
  - Simd256\_impl< true, true, false, 4 >, [646](#), [649](#), [652](#)
  - Simd256\_impl< true, true, false, 8 >, [663](#), [666](#)
  - Simd256\_impl< true, true, true, 2 >, [673](#)
  - Simd256\_impl< true, true, true, 4 >, [683](#), [689](#)
  - Simd256\_impl< true, true, true, 8 >, [699](#)
  - Simd512\_impl< true, false, true, 8 >, [708](#)
- Simd512\_impl< true, true, false, 8 >, [716](#), [719](#)
- Simd512\_impl< true, true, true, 8 >, [727](#)
- strmm\_
  - config-blas.h, [815](#)
- strsm\_
  - config-blas.h, [815](#)
- sub
  - FFLAS::vectorised, [281](#)
  - FieldSimd< \_Field >, [449](#)
  - RNSIntegerMod< RNS >, [549](#)
  - ScalFunctions< Element >, [555](#)
  - Simd128\_impl< true, true, false, 2 >, [572](#)
  - Simd128\_impl< true, true, false, 4 >, [582](#)
  - Simd128\_impl< true, true, false, 8 >, [592](#)
  - Simd128\_impl< true, true, true, 2 >, [600](#)
  - Simd128\_impl< true, true, true, 4 >, [608](#)
  - Simd128\_impl< true, true, true, 8 >, [618](#)
  - Simd256\_impl< true, false, true, 8 >, [628](#)
  - Simd256\_impl< true, true, false, 2 >, [639](#)
  - Simd256\_impl< true, true, false, 4 >, [656](#)
  - Simd256\_impl< true, true, false, 8 >, [667](#)
  - Simd256\_impl< true, true, true, 2 >, [675](#)
  - Simd256\_impl< true, true, true, 4 >, [685](#), [692](#)
  - Simd256\_impl< true, true, true, 8 >, [701](#)
  - Simd512\_impl< true, false, true, 8 >, [710](#)
  - Simd512\_impl< true, true, false, 8 >, [721](#)
  - Simd512\_impl< true, true, true, 8 >, [729](#)
- sub\_r
  - FieldSimd< \_Field >, [449](#), [450](#)
- subin
  - FieldSimd< \_Field >, [449](#)
  - ScalFunctions< Element >, [555](#)
  - Simd128\_impl< true, true, false, 2 >, [572](#)
  - Simd128\_impl< true, true, false, 4 >, [582](#)
  - Simd128\_impl< true, true, false, 8 >, [592](#)
  - Simd128\_impl< true, true, true, 2 >, [600](#)
  - Simd128\_impl< true, true, true, 4 >, [609](#)
  - Simd128\_impl< true, true, true, 8 >, [618](#)
  - Simd256\_impl< true, false, true, 8 >, [628](#)
  - Simd256\_impl< true, true, false, 2 >, [639](#)
  - Simd256\_impl< true, true, false, 4 >, [656](#)
  - Simd256\_impl< true, true, false, 8 >, [667](#)
  - Simd256\_impl< true, true, true, 2 >, [675](#)
  - Simd256\_impl< true, true, true, 4 >, [685](#), [692](#)
  - Simd256\_impl< true, true, true, 8 >, [701](#)
  - Simd512\_impl< true, false, true, 8 >, [710](#)
  - Simd512\_impl< true, true, false, 8 >, [721](#)
  - Simd512\_impl< true, true, true, 8 >, [729](#)
- subin\_r
  - FieldSimd< \_Field >, [450](#)
- subp
  - FFLAS::vectorised, [280](#)
- support\_fast\_mod< double >, [759](#)
- support\_fast\_mod< float >, [760](#)
- support\_fast\_mod< int64\_t >, [760](#)
- support\_fast\_mod< T >, [759](#)
- support\_simd< T >, [760](#)
- support\_simd\_add< T >, [761](#)

support\_simd\_mod< T >, 761

swapval

FFPACK, 383

SYNCH\_GROUP

parallel.h, 1033

SysTimer

FFLAS, 74

## T

limits< char >, 488

limits< double >, 489

limits< float >, 489

limits< Givaro::Integer >, 490

limits< int >, 490

limits< long >, 491

limits< long long >, 492

limits< Reclnt::rint< K > >, 492

limits< Reclnt::ruint< K > >, 493

limits< short int >, 494

limits< signed char >, 494

limits< unsigned char >, 495

limits< unsigned int >, 495

limits< unsigned long >, 496

limits< unsigned long long >, 497

limits< unsigned short int >, 497

## t

Simd128\_impl< true, true, false, 2 >::Converter, 425

Simd128\_impl< true, true, false, 4 >::Converter, 425

Simd128\_impl< true, true, false, 8 >::Converter, 426

Simd128\_impl< true, true, true, 2 >::Converter, 426

Simd128\_impl< true, true, true, 4 >::Converter, 426

Simd128\_impl< true, true, true, 8 >::Converter, 427

Simd256\_impl< true, false, true, 8 >::Converter, 427

Simd256\_impl< true, true, false, 2 >::Converter, 427

Simd256\_impl< true, true, false, 4 >::Converter, 428

Simd256\_impl< true, true, false, 8 >::Converter, 428

Simd256\_impl< true, true, true, 2 >::Converter, 428

Simd256\_impl< true, true, true, 4 >::Converter, 429

Simd256\_impl< true, true, true, 8 >::Converter, 429

Simd512\_impl< true, true, false, 8 >::Converter, 429

Simd512\_impl< true, true, true, 8 >::Converter, 430

## TASK

parallel.h, 1033

## tBC

test-lu.C, 1091

test-permutations.C, 1096

## Test

Test< Elt >, 763

## test

test-maxdelayeddim.C, 1092

Test< Elt >, 761

\_mm, 764

\_nn, 764

cardinality, 763

doTests, 763

Elt\_ptr, 762

enable\_if\_no\_simd\_t, 762

enable\_if\_simd128\_t, 762

enable\_if\_simd256\_t, 762

enable\_if\_simd512\_t, 763

enable\_if\_t, 762

F, 764

Field, 762

is\_same\_element, 762

Residu, 762

run, 763

Test, 763

test\_ftranspose, 763

test-charpoly-check.C, 1054

ENABLE\_CHECKER\_charpoly, 1054

main, 1055

printPolynomial, 1055

TIME\_CHECKER\_CHARPOLY, 1054

test-charpoly.C, 1055

launch\_test, 1055

main, 1056

run\_with\_field, 1055

test-compressQ.C, 1056

Field, 1056

main, 1056

printvect, 1056

test-det-check.C, 1057

ENABLE\_CHECKER\_Det, 1057

main, 1057

TIME\_CHECKER\_Det, 1057

test-det.C, 1057

main, 1058

test\_det, 1058

test-echelon.C, 1058

\_\_FFLASFFPACK\_GAUSSJORDAN\_BASECASE, 1059

\_\_FFLASFFPACK\_PLUQ\_THRESHOLD, 1059

\_\_FFLASFFPACK\_SEQUENTIAL, 1059

main, 1060

run\_with\_field, 1060

test\_colechelon, 1059

test\_redcoechelon, 1059

test\_redrowechelon, 1059

test\_rowechelon, 1059

test-fadd.C, 1060

main, 1061

test\_fadd, 1061

- test\_faddin, 1061
- test\_fsub, 1061
- test\_fsubin, 1061
- test-fdot.C, 1061
  - check\_fdot, 1062
  - ENABLE\_ALL\_CHECKINGS, 1062
  - main, 1062
  - run\_with\_field, 1062
  - run\_with\_Integer, 1062
- test-fgemm-check.C, 1063
  - ENABLE\_ALL\_CHECKINGS, 1063
  - launch\_MM\_dispatch, 1063
  - main, 1064
  - run\_with\_field, 1063
- test-fgemm.C, 1064
  - check\_MM, 1065
  - ENABLE\_CHECKER\_fgemm, 1065
  - launch\_MM, 1065
  - launch\_MM\_dispatch, 1065
  - main, 1066
  - run\_with\_field, 1066
- test-fgemv.C, 1066
  - check\_MV, 1067
  - launch\_MV, 1067
  - launch\_MV\_dispatch, 1067
  - main, 1068
  - run\_with\_field, 1067
- test-fger.C, 1068
  - check\_fger, 1069
  - launch\_fger, 1069
  - launch\_fger\_dispatch, 1069
  - main, 1069
  - run\_with\_field, 1069
  - TIME, 1068
- test-fgesv.C, 1070
  - main, 1071
  - run\_with\_field, 1070
  - test\_rect\_fgesv, 1070
  - test\_square\_fgesv, 1070
- test-finit.C, 1071
  - main, 1072
  - run\_with\_field, 1071
  - test\_freduce, 1071
- test-fscal.C, 1072
  - main, 1073
  - test\_fscal, 1072
  - test\_fscaln, 1073
- test-fsyr2k.C, 1073
  - check\_fsyr2k, 1074
  - ENABLE\_ALL\_CHECKINGS, 1074
  - main, 1074
  - run\_with\_field, 1074
- test-fsyrk.C, 1074
  - check\_computeS1S2, 1076
  - check\_fsyrk, 1075
  - check\_fsyrk\_bkdiag, 1076
  - check\_fsyrk\_diag, 1075
  - ENABLE\_ALL\_CHECKINGS, 1075
  - main, 1076
  - run\_with\_field, 1076
- test-fsytrf.C, 1076
  - main, 1078
  - operator<<, 1077
  - run\_with\_field, 1077
  - test\_generic\_fsytrf, 1077
  - test\_RPM\_fsytrf, 1077
- test-ftrmm.C, 1078
  - \_\_FFLASFFPACK\_SEQUENTIAL, 1078
  - check\_ftrmm, 1078
  - main, 1079
  - run\_with\_field, 1079
- test-ftrmv.C, 1079
  - \_\_FFLASFFPACK\_SEQUENTIAL, 1079
  - check\_ftrmv, 1080
  - ENABLE\_ALL\_CHECKINGS, 1079
  - main, 1080
  - run\_with\_field, 1080
- test-ftrsm-check.C, 1080
  - ENABLE\_ALL\_CHECKINGS, 1080
  - main, 1080
- test-ftrsm.C, 1081
  - \_\_FFLASFFPACK\_SEQUENTIAL, 1081
  - check\_ftrsm, 1081
  - ENABLE\_ALL\_CHECKINGS, 1081
  - main, 1082
  - run\_with\_field, 1081
- test-ftrssyr2k.C, 1082
  - check\_ftrssyr2k, 1082
  - ENABLE\_ALL\_CHECKINGS, 1082
  - main, 1083
  - run\_with\_field, 1083
- test-ftrstr.C, 1083
  - check\_ftrstr, 1084
  - ENABLE\_ALL\_CHECKINGS, 1083
  - main, 1084
  - run\_with\_field, 1084
- test-ftrsv.C, 1084
  - \_\_FFLASFFPACK\_SEQUENTIAL, 1085
  - check\_ftrsv, 1085
  - ENABLE\_ALL\_CHECKINGS, 1085
  - main, 1085
  - run\_with\_field, 1085
- test-ftrtri.C, 1085
  - \_\_FFLASFFPACK\_SEQUENTIAL, 1086
  - check\_ftrtri, 1086
  - ENABLE\_ALL\_CHECKINGS, 1086
  - main, 1086
  - run\_with\_field, 1086
- test-interfaces-c.c, 1086
  - main, 1087
- test-invert-check.C, 1087
  - ENABLE\_ALL\_CHECKINGS, 1087
  - main, 1087
- test-io.C, 1087
  - main, 1088
  - run\_with\_field, 1088

- test-lu.C, 1088
  - \_\_FFLASFFPACK\_SEQUENTIAL, 1089
  - \_\_LUDIVINE\_CUTOFF, 1089
  - BASECASE\_K, 1089
  - launch\_test, 1091
  - main, 1091
  - mvcnt, 1092
  - run\_with\_field, 1091
  - tBC, 1091
  - test\_LUdivine, 1089
  - test\_pluq, 1090
  - tgemm, 1091
  - timtot, 1092
  - tperm, 1091
  - trest, 1092
  - ttrsm, 1092
  - verifPLUQ, 1090
- test-maxdelayeddim.C, 1092
  - main, 1092
  - MAX\_WITH\_SIZE\_T, 1092
  - test, 1092
- test-minpoly.C, 1093
  - check\_minpoly, 1093
  - main, 1093
  - run\_with\_field, 1093
- test-multifile1.C, 1093
- test-multifile2.C, 1094
  - main, 1094
- test-nullspace.C, 1094
  - checkingMessage, 1094
  - main, 1095
  - readOrRandomMatrixWithRankAndRandomRPM, 1094
  - run\_with\_field, 1095
  - test\_nullspace, 1095
- test-permutations.C, 1095
  - checkMonotonicApplyP, 1096
  - main, 1096
  - tBC, 1096
  - tgemm, 1096
  - timtot, 1096
  - tperm, 1096
  - trest, 1096
  - ttrsm, 1096
- test-pluq-check.C, 1096
  - ENABLE\_ALL\_CHECKINGS, 1097
  - main, 1097
- test-quasisep.C, 1097
  - launch\_test, 1097
  - main, 1098
  - run\_with\_field, 1098
  - test\_BruhatGenerator, 1097
  - testLTQSRPM, 1098
- test-rankprofiles.C, 1098
  - \_\_FFLASFFPACK\_SEQUENTIAL, 1099
  - main, 1099
  - run\_with\_field, 1099
- test-rpm.C, 1099
  - checkRPM, 1099
  - checkSymmetricRPM, 1099
  - main, 1100
- test-simd.C, 1100
  - \_TEST\_ONE, 1101
  - check\_eq, 1102
  - cmp, 1102
  - eval\_func\_on\_array, 1102
  - main, 1103
  - operator<<, 1102
  - TEST\_IMPL, 1101
  - test\_impl, 1103
  - test\_impl\_base, 1102, 1103
  - TEST\_ONE\_OP, 1101
  - TEST\_ONE\_OP\_WZ, 1101
- test-solve.C, 1103
  - check\_solve, 1103
  - main, 1104
  - run\_with\_field, 1103
- test-storage-transpose.C, 1104
  - main, 1104
- test-utils.h, 1049
- test\_BruhatGenerator
  - test-quasisep.C, 1097
- test\_colechelon
  - test-echelon.C, 1059
- test\_det
  - test-det.C, 1058
- test\_fadd
  - test-fadd.C, 1061
- test\_faddin
  - test-fadd.C, 1061
- test\_freduce
  - test-finit.C, 1071
- test\_fscal
  - test-fscal.C, 1072
- test\_fscalin
  - test-fscal.C, 1073
- test\_fsub
  - test-fadd.C, 1061
- test\_fsubin
  - test-fadd.C, 1061
- test\_ftranspose
  - Test< Elt >, 763
- test\_generic\_fsytrf
  - test-fsytrf.C, 1077
- TEST\_IMPL
  - test-simd.C, 1101
- test\_impl
  - test-simd.C, 1103
- test\_impl\_base
  - test-simd.C, 1102, 1103
- test\_LUdivine
  - test-lu.C, 1089
- test\_nullspace
  - test-nullspace.C, 1095
- TEST\_ONE\_OP
  - test-simd.C, 1101



- TEST\_ONE\_OP\_WZ
  - test-simd.C, [1101](#)
- test\_pluq
  - test-lu.C, [1090](#)
- test\_rect\_fgesv
  - test-fgesv.C, [1070](#)
- test\_redcoechelon
  - test-echelon.C, [1059](#)
- test\_redrowechelon
  - test-echelon.C, [1059](#)
- test\_rowechelon
  - test-echelon.C, [1059](#)
- test\_RPM\_fsytrf
  - test-fsytrf.C, [1077](#)
- test\_square\_fgesv
  - test-fgesv.C, [1070](#)
- testLTQSRPM
  - test-quasisep.C, [1098](#)
- TestOneMethod
  - TestOneMethod< Simd >, [765](#)
- TestOneMethod< Simd >, [764](#)
  - Element, [765](#)
  - enable\_if\_t, [765](#)
  - evaluate\_scalar\_method, [765](#), [766](#)
  - evaluate\_simd\_method, [766](#)
  - getStatus, [766](#)
  - getTestName, [766](#)
  - inputs, [767](#)
  - name, [767](#)
  - nb\_lref, [766](#)
  - outputs\_scalar, [767](#)
  - outputs\_simd, [767](#)
  - TestOneMethod, [765](#)
  - vect\_size, [766](#)
  - vect\_t, [765](#)
  - vectElt, [765](#)
  - writeDebugData, [766](#)
  - writeResultLine, [766](#)
- tfn\_minus, [767](#)
  - operator(), [767](#)
- tfn\_minus\_eq, [767](#)
  - operator(), [768](#)
- tfn\_mul, [768](#)
  - operator(), [768](#)
- tfn\_mul\_eq, [768](#)
  - operator(), [768](#)
- tfn\_plus, [768](#)
  - operator(), [769](#)
- tfn\_plus\_eq, [769](#)
  - operator(), [769](#)
- tgemm
  - test-lu.C, [1091](#)
  - test-permutations.C, [1096](#)
- THREAD\_INDEX
  - parallel.h, [1033](#)
- THREADS
  - benchmark-fgemm-rns.C, [788](#)
- Threads, [769](#)
- threads\_fgemm
  - FFPACK, [363](#)
- threads\_ftsm
  - FFPACK, [363](#)
- THREED
  - benchmark-fgemm-rns.C, [788](#)
- ThreeD, [769](#)
- THREEDA
  - benchmark-fgemm-rns.C, [788](#)
- ThreeDAdaptive, [769](#)
- ThreeDInPlace, [770](#)
- THREEDIP
  - benchmark-fgemm-rns.C, [788](#)
- TIME
  - test-fger.C, [1068](#)
- TIME\_CHECKER\_CHARPOLY
  - test-charpoly-check.C, [1054](#)
- TIME\_CHECKER\_Det
  - test-det-check.C, [1057](#)
- Timer
  - FFLAS, [74](#)
- timer.h, [1050](#)
- timtot
  - test-lu.C, [1092](#)
  - test-permutations.C, [1096](#)
- TInverter
  - FFPACK, [345](#), [348](#)
- tmain
  - benchmark-fgemm-mp.C, [787](#)
  - benchmark-fgemv-mp.C, [790](#)
- Todo List, [9](#)
- tperm
  - test-lu.C, [1091](#)
  - test-permutations.C, [1096](#)
- transpose
  - BlockTransposeSIMD< Field, Simd, >, [410](#)
  - Simd128\_impl< true, true, false, 2 >, [571](#)
  - Simd128\_impl< true, true, false, 4 >, [581](#)
  - Simd128\_impl< true, true, false, 8 >, [591](#)
  - Simd128\_impl< true, true, true, 2 >, [599](#)
  - Simd128\_impl< true, true, true, 4 >, [608](#)
  - Simd128\_impl< true, true, true, 8 >, [617](#)
  - Simd256\_impl< true, false, true, 8 >, [627](#)
  - Simd256\_impl< true, true, false, 2 >, [639](#)
  - Simd256\_impl< true, true, false, 4 >, [655](#)
  - Simd256\_impl< true, true, false, 8 >, [667](#)
  - Simd256\_impl< true, true, true, 2 >, [674](#)
  - Simd256\_impl< true, true, true, 4 >, [685](#), [691](#)
  - Simd256\_impl< true, true, true, 8 >, [700](#)
  - Simd512\_impl< true, false, true, 8 >, [709](#)
  - Simd512\_impl< true, true, false, 8 >, [720](#)
  - Simd512\_impl< true, true, true, 8 >, [728](#)
- trest
  - test-lu.C, [1092](#)
  - test-permutations.C, [1096](#)
- trinv\_left
  - FFPACK, [310](#), [367](#)
- trinv\_left\_modular\_double



- ffpack.C, [991](#)
- ffpack\_c.h, [1013](#)
- TRSMBound
  - FFLAS::Protected, [212](#), [213](#)
- TRSMHelper
  - TRSMHelper< ReclterTrait, ParSeqTrait >, [770](#)
- TRSMHelper< ReclterTrait, ParSeqTrait >, [770](#)
  - parseq, [771](#)
  - pMMH, [771](#)
  - TRSMHelper, [770](#)
- TTimer
  - arithprog.C, [773](#)
  - benchmark-dgemm.C, [782](#)
  - charpoly.C, [774](#)
- ttrsm
  - test-lu.C, [1092](#)
  - test-permutations.C, [1096](#)
- Tutorial, [2](#)
- TWOD
  - benchmark-fgemm-rns.C, [788](#)
- TwoD, [771](#)
- TWODA
  - benchmark-fgemm-rns.C, [788](#)
- TwoDAdaptive, [771](#)
- type
  - Argument, [404](#)
  - associatedDelayedField< const FFPACK::RNSIntegerModRNS >, [404](#)
  - associatedDelayedField< const Givaro::Modular< T, X > >, [405](#)
  - associatedDelayedField< const Givaro::ModularBalanced< T > >, [405](#)
  - associatedDelayedField< const Givaro::ZRing< T > >, [406](#)
  - associatedDelayedField< Field >, [404](#)
  - CompactElement< double >, [422](#)
  - CompactElement< Element >, [421](#)
  - CompactElement< float >, [422](#)
  - CompactElement< int16\_t >, [422](#)
  - CompactElement< int32\_t >, [422](#)
  - CompactElement< int64\_t >, [423](#)
  - is\_simd< T >, [481](#)
- TYPE\_BOOL
  - args-parser.h, [1041](#)
- TYPE\_DOUBLE
  - args-parser.h, [1041](#)
- TYPE\_INT
  - args-parser.h, [1041](#)
- TYPE\_INTEGER
  - args-parser.h, [1041](#)
- type\_integer
  - args-parser.h, [1041](#)
- TYPE\_INTLIST
  - args-parser.h, [1041](#)
- TYPE\_LONGLONG
  - args-parser.h, [1041](#)
- TYPE\_NONE
  - args-parser.h, [1041](#)
- TYPE\_STR
  - args-parser.h, [1041](#)
- type\_string
  - NoSimd< T >, [521](#)
  - Simd128\_impl< true, true, false, 2 >, [567](#)
  - Simd128\_impl< true, true, false, 4 >, [577](#)
  - Simd128\_impl< true, true, false, 8 >, [587](#)
  - Simd128\_impl< true, true, true, 2 >, [597](#)
  - Simd128\_impl< true, true, true, 4 >, [606](#)
  - Simd128\_impl< true, true, true, 8 >, [615](#)
  - Simd256\_impl< true, false, true, 8 >, [625](#)
  - Simd256\_impl< true, true, false, 2 >, [634](#)
  - Simd256\_impl< true, true, false, 4 >, [645](#), [648](#)
  - Simd256\_impl< true, true, false, 8 >, [662](#)
  - Simd256\_impl< true, true, true, 2 >, [672](#)
  - Simd256\_impl< true, true, true, 4 >, [682](#), [688](#)
  - Simd256\_impl< true, true, true, 8 >, [698](#)
  - Simd512\_impl< true, false, true, 8 >, [707](#)
  - Simd512\_impl< true, true, false, 8 >, [715](#)
  - Simd512\_impl< true, true, true, 8 >, [726](#)
- TYPE\_UINT64
  - args-parser.h, [1041](#)
- unfit
  - FFLAS::Protected, [221](#)
- unpackhi
  - ScalFunctions< Element >, [557](#)
  - Simd128\_impl< true, true, false, 2 >, [571](#)
  - Simd128\_impl< true, true, false, 4 >, [581](#)
  - Simd128\_impl< true, true, false, 8 >, [591](#)
  - Simd128\_impl< true, true, true, 2 >, [599](#)
  - Simd128\_impl< true, true, true, 4 >, [607](#)
  - Simd128\_impl< true, true, true, 8 >, [617](#)
  - Simd256\_impl< true, false, true, 8 >, [627](#)
  - Simd256\_impl< true, true, false, 2 >, [638](#)
  - Simd256\_impl< true, true, false, 4 >, [653](#)
  - Simd256\_impl< true, true, false, 8 >, [666](#)
  - Simd256\_impl< true, true, true, 2 >, [674](#)
  - Simd256\_impl< true, true, true, 4 >, [684](#), [690](#)
  - Simd256\_impl< true, true, true, 8 >, [700](#)
  - Simd512\_impl< true, false, true, 8 >, [709](#)
  - Simd512\_impl< true, true, false, 8 >, [720](#)
  - Simd512\_impl< true, true, true, 8 >, [728](#)
- unpackhi\_intrinsic
  - Simd128\_impl< true, true, false, 2 >, [571](#)
  - Simd128\_impl< true, true, false, 4 >, [580](#)
  - Simd128\_impl< true, true, false, 8 >, [591](#)
  - Simd128\_impl< true, true, true, 2 >, [598](#)
  - Simd128\_impl< true, true, true, 4 >, [607](#)
  - Simd128\_impl< true, true, true, 8 >, [616](#)
  - Simd256\_impl< true, false, true, 8 >, [627](#)
  - Simd256\_impl< true, true, false, 2 >, [638](#)
  - Simd256\_impl< true, true, false, 4 >, [653](#)
  - Simd256\_impl< true, true, false, 8 >, [666](#)
  - Simd256\_impl< true, true, true, 2 >, [673](#)
  - Simd256\_impl< true, true, true, 4 >, [684](#), [690](#)
  - Simd256\_impl< true, true, true, 8 >, [700](#)
  - Simd512\_impl< true, false, true, 8 >, [708](#)
  - Simd512\_impl< true, true, false, 8 >, [720](#)

- Simd512\_impl< true, true, true, 8 >, [728](#)
- unpacklo
  - ScalFunctions< Element >, [557](#)
  - Simd128\_impl< true, true, false, 2 >, [571](#)
  - Simd128\_impl< true, true, false, 4 >, [581](#)
  - Simd128\_impl< true, true, false, 8 >, [591](#)
  - Simd128\_impl< true, true, true, 2 >, [598](#)
  - Simd128\_impl< true, true, true, 4 >, [607](#)
  - Simd128\_impl< true, true, true, 8 >, [617](#)
  - Simd256\_impl< true, false, true, 8 >, [627](#)
  - Simd256\_impl< true, true, false, 2 >, [638](#)
  - Simd256\_impl< true, true, false, 4 >, [653](#)
  - Simd256\_impl< true, true, false, 8 >, [666](#)
  - Simd256\_impl< true, true, true, 2 >, [674](#)
  - Simd256\_impl< true, true, true, 4 >, [684](#), [690](#)
  - Simd256\_impl< true, true, true, 8 >, [700](#)
  - Simd512\_impl< true, false, true, 8 >, [709](#)
  - Simd512\_impl< true, true, false, 8 >, [720](#)
  - Simd512\_impl< true, true, true, 8 >, [728](#)
- unpacklo\_intrinsic
  - Simd128\_impl< true, true, false, 2 >, [570](#)
  - Simd128\_impl< true, true, false, 4 >, [580](#)
  - Simd128\_impl< true, true, false, 8 >, [590](#)
  - Simd128\_impl< true, true, true, 2 >, [598](#)
  - Simd128\_impl< true, true, true, 4 >, [607](#)
  - Simd128\_impl< true, true, true, 8 >, [616](#)
  - Simd256\_impl< true, false, true, 8 >, [627](#)
  - Simd256\_impl< true, true, false, 2 >, [638](#)
  - Simd256\_impl< true, true, false, 4 >, [653](#)
  - Simd256\_impl< true, true, false, 8 >, [666](#)
  - Simd256\_impl< true, true, true, 2 >, [673](#)
  - Simd256\_impl< true, true, true, 4 >, [684](#), [690](#)
  - Simd256\_impl< true, true, true, 8 >, [699](#)
  - Simd512\_impl< true, false, true, 8 >, [708](#)
  - Simd512\_impl< true, true, false, 8 >, [719](#)
  - Simd512\_impl< true, true, true, 8 >, [728](#)
- unpacklohi
  - ScalFunctions< Element >, [558](#)
  - Simd128\_impl< true, true, false, 2 >, [571](#)
  - Simd128\_impl< true, true, false, 4 >, [581](#)
  - Simd128\_impl< true, true, false, 8 >, [591](#)
  - Simd128\_impl< true, true, true, 2 >, [599](#)
  - Simd128\_impl< true, true, true, 4 >, [608](#)
  - Simd128\_impl< true, true, true, 8 >, [617](#)
  - Simd256\_impl< true, false, true, 8 >, [627](#)
  - Simd256\_impl< true, true, false, 2 >, [638](#)
  - Simd256\_impl< true, true, false, 4 >, [654](#)
  - Simd256\_impl< true, true, false, 8 >, [666](#)
  - Simd256\_impl< true, true, true, 2 >, [674](#)
  - Simd256\_impl< true, true, true, 4 >, [684](#), [690](#)
  - Simd256\_impl< true, true, true, 8 >, [700](#)
  - Simd512\_impl< true, false, true, 8 >, [709](#)
  - Simd512\_impl< true, true, false, 8 >, [720](#)
  - Simd512\_impl< true, true, true, 8 >, [728](#)
- UnparametricTag, [771](#)
- updateD
  - FFPACK::Protected, [396](#)
- USE\_OPENMP
- config.h, [820](#)
- UserTimer
  - FFLAS, [74](#)
- utils.h, [913](#)
- v
  - Simd128\_impl< true, true, false, 2 >::Converter, [425](#)
  - Simd128\_impl< true, true, false, 4 >::Converter, [425](#)
  - Simd128\_impl< true, true, false, 8 >::Converter, [426](#)
  - Simd128\_impl< true, true, true, 2 >::Converter, [426](#)
  - Simd128\_impl< true, true, true, 4 >::Converter, [426](#)
  - Simd128\_impl< true, true, true, 8 >::Converter, [427](#)
  - Simd256\_impl< true, false, true, 8 >::Converter, [427](#)
  - Simd256\_impl< true, true, false, 2 >::Converter, [427](#)
  - Simd256\_impl< true, true, false, 4 >::Converter, [428](#)
  - Simd256\_impl< true, true, false, 8 >::Converter, [428](#)
  - Simd256\_impl< true, true, true, 2 >::Converter, [428](#)
  - Simd256\_impl< true, true, true, 4 >::Converter, [429](#)
  - Simd256\_impl< true, true, true, 8 >::Converter, [429](#)
  - Simd512\_impl< true, true, false, 8 >::Converter, [429](#)
  - Simd512\_impl< true, true, true, 8 >::Converter, [430](#)
- val
  - Coo< Field >, [433](#)
  - Coo< ValT, IdxT >, [431](#), [434](#)
- valid
  - NoSimd< T >, [521](#)
  - Simd128\_impl< true, true, false, 2 >, [569](#)
  - Simd128\_impl< true, true, false, 4 >, [579](#)
  - Simd128\_impl< true, true, false, 8 >, [589](#)
  - Simd128\_impl< true, true, true, 2 >, [597](#)
  - Simd128\_impl< true, true, true, 4 >, [606](#)
  - Simd128\_impl< true, true, true, 8 >, [615](#)
  - Simd256\_impl< true, false, true, 8 >, [625](#)
  - Simd256\_impl< true, true, false, 2 >, [636](#)
  - Simd256\_impl< true, true, false, 4 >, [650](#)
  - Simd256\_impl< true, true, false, 8 >, [665](#)
  - Simd256\_impl< true, true, true, 2 >, [672](#)
  - Simd256\_impl< true, true, true, 4 >, [682](#), [688](#)
  - Simd256\_impl< true, true, true, 8 >, [698](#)
  - Simd512\_impl< true, false, true, 8 >, [707](#)
  - Simd512\_impl< true, true, false, 8 >, [718](#)
  - Simd512\_impl< true, true, true, 8 >, [726](#)
- VALUE
  - parallel.h, [1034](#)

## value

AlgoChooser< ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq >, [401](#)  
 AlgoChooser< ModeT, ParSeq >, [401](#)  
 ALL< false, v... >, [402](#)  
 ALL< true, v... >, [402](#)  
 ALL<>, [402](#)  
 AreEqual< X, X >, [403](#)  
 AreEqual< X, Y >, [403](#)  
 compatible\_data\_type< Field >, [423](#)  
 compatible\_data\_type< Givaro::ZRing< double > >, [423](#)  
 compatible\_data\_type< Givaro::ZRing< float > >, [423](#)  
 ElementTraits< double >, [439](#)  
 ElementTraits< Element >, [439](#)  
 ElementTraits< FFPACK::rns\_double\_elt >, [439](#)  
 ElementTraits< float >, [439](#)  
 ElementTraits< Givaro::Integer >, [440](#)  
 ElementTraits< int16\_t >, [440](#)  
 ElementTraits< int32\_t >, [440](#)  
 ElementTraits< int64\_t >, [441](#)  
 ElementTraits< int8\_t >, [441](#)  
 ElementTraits< Reclnt::rint< K > >, [441](#)  
 ElementTraits< Reclnt::rmint< K, MG > >, [441](#)  
 ElementTraits< Reclnt::ruint< K > >, [442](#)  
 ElementTraits< uint16\_t >, [442](#)  
 ElementTraits< uint32\_t >, [442](#)  
 ElementTraits< uint64\_t >, [443](#)  
 ElementTraits< uint8\_t >, [443](#)  
 has\_minus\_eq\_impl< C >, [472](#)  
 has\_minus\_impl< C >, [472](#)  
 has\_mul\_eq\_impl< C >, [472](#)  
 has\_mul\_impl< C >, [473](#)  
 has\_operation< T >, [473](#)  
 has\_plus\_eq\_impl< C >, [473](#)  
 has\_plus\_impl< C >, [474](#)  
 is\_all\_same< T, Args... >, [480](#)  
 is\_all\_same<>, [481](#)  
 is\_simd< T >, [481](#)  
 ModeTraits< Field >, [512](#)  
 ModeTraits< Givaro::Modular< Element, Compute > >, [513](#)  
 ModeTraits< Givaro::Modular< Givaro::Integer, Compute > >, [513](#)  
 ModeTraits< Givaro::Modular< int16\_t, Compute > >, [513](#)  
 ModeTraits< Givaro::Modular< int32\_t, Compute > >, [514](#)  
 ModeTraits< Givaro::Modular< int64\_t, uint64\_t > >, [514](#)  
 ModeTraits< Givaro::Modular< int8\_t, Compute > >, [514](#)  
 ModeTraits< Givaro::Modular< Reclnt::ruint< K >, Compute > >, [515](#)  
 ModeTraits< Givaro::Modular< uint16\_t, Compute > >, [515](#)

ModeTraits< Givaro::Modular< uint32\_t, Compute > >, [515](#)  
 ModeTraits< Givaro::Modular< uint8\_t, Compute > >, [516](#)  
 ModeTraits< Givaro::ModularBalanced< Element > >, [516](#)  
 ModeTraits< Givaro::ModularBalanced< Givaro::Integer > >, [516](#)  
 ModeTraits< Givaro::ModularBalanced< int16\_t > >, [517](#)  
 ModeTraits< Givaro::ModularBalanced< int32\_t > >, [517](#)  
 ModeTraits< Givaro::ModularBalanced< int8\_t > >, [517](#)  
 ModeTraits< Givaro::Montgomery< T > >, [518](#)  
 ModeTraits< Givaro::ZRing< double > >, [518](#)  
 ModeTraits< Givaro::ZRing< float > >, [518](#)  
 ModeTraits< Givaro::ZRing< Givaro::Integer > >, [519](#)  
 need\_field\_characteristic< Field >, [519](#)  
 need\_field\_characteristic< Givaro::Modular< Field > >, [520](#)  
 need\_field\_characteristic< Givaro::ModularBalanced< Field > >, [520](#)  
 SimdChooser< T, false, b >, [734](#)  
 SimdChooser< T, true, false >, [735](#)  
 SimdChooser< T, true, true >, [735](#)  
 width< double >, [772](#)  
 width< float >, [772](#)  
 width< T >, [772](#)

## vand

ScalFunctions< Element >, [555](#)  
 Simd128\_impl< true, true, false, 2 >, [574](#)  
 Simd128\_impl< true, true, false, 4 >, [583](#)  
 Simd128\_impl< true, true, false, 8 >, [594](#)  
 Simd128\_impl< true, true, true, 2 >, [603](#)  
 Simd128\_impl< true, true, true, 4 >, [612](#)  
 Simd128\_impl< true, true, true, 8 >, [621](#)  
 Simd128i\_base, [623](#)  
 Simd256\_impl< true, false, true, 8 >, [630](#)  
 Simd256\_impl< true, true, false, 4 >, [659](#)  
 Simd256\_impl< true, true, true, 4 >, [695](#)  
 Simd512\_impl< true, true, false, 8 >, [723](#)  
 Simd512\_impl< true, true, true, 8 >, [732](#)  
 Simd512i\_base, [734](#)

## vandnot

ScalFunctions< Element >, [555](#)  
 Simd128\_impl< true, true, false, 2 >, [574](#)  
 Simd128\_impl< true, true, false, 4 >, [584](#)  
 Simd128\_impl< true, true, false, 8 >, [594](#)  
 Simd128\_impl< true, true, true, 2 >, [603](#)  
 Simd128\_impl< true, true, true, 4 >, [612](#)  
 Simd128\_impl< true, true, true, 8 >, [621](#)  
 Simd128i\_base, [623](#)  
 Simd256\_impl< true, false, true, 8 >, [630](#)  
 Simd256\_impl< true, true, false, 4 >, [659](#)  
 Simd256\_impl< true, true, true, 4 >, [695](#)  
 Simd512\_impl< true, true, false, 8 >, [723](#)

- Simd512\_impl< true, true, true, 8 >, [732](#)
  - Simd512i\_base, [734](#)
- VEC\_ADD
  - FFLAS::vectorised, [280](#)
- VEC\_SUB
  - FFLAS::vectorised, [280](#)
- vect\_size
  - FieldSimd< \_Field >, [452](#)
  - NoSimd< T >, [521](#)
  - Simd128\_impl< true, true, false, 2 >, [574](#)
  - Simd128\_impl< true, true, false, 4 >, [584](#)
  - Simd128\_impl< true, true, false, 8 >, [594](#)
  - Simd128\_impl< true, true, true, 2 >, [603](#)
  - Simd128\_impl< true, true, true, 4 >, [612](#)
  - Simd128\_impl< true, true, true, 8 >, [622](#)
  - Simd256\_impl< true, false, true, 8 >, [631](#)
  - Simd256\_impl< true, true, false, 2 >, [641](#)
  - Simd256\_impl< true, true, false, 4 >, [659](#)
  - Simd256\_impl< true, true, false, 8 >, [669](#)
  - Simd256\_impl< true, true, true, 2 >, [678](#)
  - Simd256\_impl< true, true, true, 4 >, [695](#)
  - Simd256\_impl< true, true, true, 8 >, [704](#)
  - Simd512\_impl< true, false, true, 8 >, [712](#)
  - Simd512\_impl< true, true, false, 8 >, [723](#)
  - Simd512\_impl< true, true, true, 8 >, [733](#)
  - TestOneMethod< Simd >, [766](#)
- vect\_t
  - FieldSimd< \_Field >, [447](#)
  - NoSimd< T >, [520](#)
  - Simd128\_impl< true, true, false, 2 >, [566](#)
  - Simd128\_impl< true, true, false, 4 >, [577](#)
  - Simd128\_impl< true, true, false, 8 >, [587](#)
  - Simd128\_impl< true, true, true, 2 >, [596](#)
  - Simd128\_impl< true, true, true, 4 >, [605](#)
  - Simd128\_impl< true, true, true, 8 >, [614](#)
  - simd128\_int64.inl, [876](#)
  - Simd128i\_base, [622](#)
  - Simd256\_impl< true, false, true, 8 >, [625](#)
  - Simd256\_impl< true, true, false, 2 >, [634](#)
  - Simd256\_impl< true, true, false, 4 >, [645](#)
  - Simd256\_impl< true, true, false, 8 >, [662](#)
  - Simd256\_impl< true, true, true, 2 >, [671](#)
  - Simd256\_impl< true, true, true, 4 >, [682](#)
  - Simd256\_impl< true, true, true, 8 >, [697](#)
  - simd256\_int64.inl, [878](#)
  - Simd256i\_base, [705](#)
  - Simd512\_impl< true, false, true, 8 >, [707](#)
  - Simd512\_impl< true, true, false, 8 >, [715](#)
  - Simd512\_impl< true, true, true, 8 >, [725](#)
  - simd512\_int64.inl, [881](#)
  - Simd512i\_base, [733](#)
  - TestOneMethod< Simd >, [765](#)
- vectElt
  - ScalFunctions< Element >, [554](#)
  - TestOneMethod< Simd >, [765](#)
- verification\_PLUQ
  - benchmark-pluq.C, [801](#)
- verifPLUQ
  - test-lu.C, [1090](#)
- VERSION
  - config.h, [820](#)
- vor
  - ScalFunctions< Element >, [555](#)
  - Simd128\_impl< true, true, false, 2 >, [574](#)
  - Simd128\_impl< true, true, false, 4 >, [584](#)
  - Simd128\_impl< true, true, false, 8 >, [594](#)
  - Simd128\_impl< true, true, true, 2 >, [603](#)
  - Simd128\_impl< true, true, true, 4 >, [612](#)
  - Simd128\_impl< true, true, true, 8 >, [621](#)
  - Simd128i\_base, [623](#)
  - Simd256\_impl< true, false, true, 8 >, [630](#)
  - Simd256\_impl< true, true, false, 4 >, [659](#)
  - Simd256\_impl< true, true, true, 4 >, [695](#)
  - Simd512\_impl< true, true, false, 8 >, [723](#)
  - Simd512\_impl< true, true, true, 8 >, [732](#)
  - Simd512i\_base, [733](#)
- wxor
  - ScalFunctions< Element >, [555](#)
  - Simd128\_impl< true, true, false, 2 >, [574](#)
  - Simd128\_impl< true, true, false, 4 >, [584](#)
  - Simd128\_impl< true, true, false, 8 >, [594](#)
  - Simd128\_impl< true, true, true, 2 >, [603](#)
  - Simd128\_impl< true, true, true, 4 >, [612](#)
  - Simd128\_impl< true, true, true, 8 >, [621](#)
  - Simd128i\_base, [623](#)
  - Simd256\_impl< true, false, true, 8 >, [630](#)
  - Simd256\_impl< true, true, false, 4 >, [659](#)
  - Simd256\_impl< true, true, true, 4 >, [695](#)
  - Simd512\_impl< true, true, false, 8 >, [723](#)
  - Simd512\_impl< true, true, true, 8 >, [732](#)
  - Simd512i\_base, [733](#)
- WAIT
  - parallel.h, [1033](#)
- width< double >, [772](#)
  - value, [772](#)
- width< float >, [772](#)
  - value, [772](#)
- width< T >, [772](#)
  - value, [772](#)
- Winograd, [772](#)
  - FFLAS::BLAS3, [193](#)
- winograd.C, [778](#)
  - balanced, [779](#)
  - DOUBLE\_TO\_FLOAT\_CROSSOVER, [778](#)
  - GFOPS, [778](#)
  - main, [779](#)
- Winograd\_L\_S
  - FFLAS::BLAS3, [196](#)
- Winograd\_LR\_S
  - FFLAS::BLAS3, [196](#)
- Winograd\_R\_S
  - FFLAS::BLAS3, [197](#)
- WinogradAcc\_2\_24
  - FFLAS::BLAS3, [194](#)
- WinogradAcc\_2\_27
  - FFLAS::BLAS3, [194](#)

- WinogradAcc\_3\_21
  - FFLAS::BLAS3, [194](#)
- WinogradAcc\_3\_23
  - FFLAS::BLAS3, [193](#)
- WinogradAcc\_L\_S
  - FFLAS::BLAS3, [196](#)
- WinogradAcc\_LR
  - FFLAS::BLAS3, [195](#)
- WinogradAcc\_R\_S
  - FFLAS::BLAS3, [195](#)
- WinogradCalc
  - FFLAS::Protected, [218](#)
- WinogradPar, [772](#)
- WinogradSteps
  - FFLAS::Protected, [216](#)
- WinogradThreshold
  - FFLAS::Protected, [216](#)
- WinoPar
  - FFLAS::BLAS3, [193](#)
- WINOTHRESHOLD
  - fflas.h, [826](#)
- WRITE
  - parallel.h, [1034](#)
- write
  - RNSInteger< RNS >, [545](#)
  - RNSIntegerMod< RNS >, [550](#)
- write\_field
  - Matio.h, [1049](#)
- write\_matrix
  - benchmark-fgemv-mp.C, [790](#)
  - RNSIntegerMod< RNS >, [550](#)
- write\_matrix\_long
  - RNSIntegerMod< RNS >, [551](#)
- writeCommandString
  - FFLAS, [185](#)
- writeDebugData
  - TestOneMethod< Simd >, [766](#)
- writeDnsFormat
  - FFLAS, [152](#)
- WriteMatrix
  - FFLAS, [185](#), [187](#)
- WritePermutation
  - FFLAS, [187](#)
- writeResultLine
  - TestOneMethod< Simd >, [766](#)
- zero
  - FFLAS, [76](#)
  - FieldSimd< \_Field >, [450](#)
  - RNSInteger< RNS >, [545](#)
  - RNSIntegerMod< RNS >, [552](#)
  - ScalFunctions< Element >, [555](#)
  - Simd128\_impl< true, true, false, 2 >, [573](#)
  - Simd128\_impl< true, true, false, 4 >, [583](#)
  - Simd128\_impl< true, true, false, 8 >, [593](#)
  - Simd128\_impl< true, true, true, 2 >, [603](#)
  - Simd128\_impl< true, true, true, 4 >, [611](#)
  - Simd128\_impl< true, true, true, 8 >, [621](#)
  - Simd128i\_base, [622](#)
  - Simd256\_impl< true, false, true, 8 >, [626](#)
  - Simd256\_impl< true, true, false, 2 >, [641](#)
  - Simd256\_impl< true, true, false, 4 >, [659](#)
  - Simd256\_impl< true, true, false, 8 >, [669](#)
  - Simd256\_impl< true, true, true, 2 >, [678](#)
  - Simd256\_impl< true, true, true, 4 >, [695](#)
  - Simd256\_impl< true, true, true, 8 >, [704](#)
  - Simd256i\_base, [705](#)
  - Simd512\_impl< true, false, true, 8 >, [707](#)
  - Simd512\_impl< true, true, false, 8 >, [723](#)
  - Simd512\_impl< true, true, true, 8 >, [732](#)
  - Simd512i\_base, [733](#)
- ZOSparseMatrix
  - FFLAS, [72](#)