

FflasFfpack

Generated on Thu Dec 12 2024 00:00:00 for FflasFfpack by Doxygen 1.13.1

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 . . . . .	42
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 . . . . .	43
10.5.1 Detailed Description . . . . .	43
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 psub()	79
11.1.3.13 pfaddin()	80
11.1.3.14 psubin()	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>ftsm()</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>ftsm()</code> [7/9]	144
11.1.3.187 <code>ftsm()</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 ftranspose()	154
11.1.3.235 maxCardinality()	154
11.1.3.236 maxCardinality< Givaro::Modular< int64_t > >()	154
11.1.3.237 maxCardinality< Givaro::Modular< int32_t > >()	154
11.1.3.238 minCardinality()	155
11.1.3.239 fflas_delete() [1/4]	155
11.1.3.240 fflas_delete() [2/4]	155
11.1.3.241 fflas_new() [1/7]	155
11.1.3.242 fflas_new() [2/7]	155
11.1.3.243 finit_rns() [1/2]	155
11.1.3.244 finit_trans_rns()	156
11.1.3.245 fconvert_rns() [1/2]	156
11.1.3.246 fconvert_trans_rns()	156
11.1.3.247 fflas_new() [3/7]	156
11.1.3.248 fflas_new() [4/7]	156
11.1.3.249 finit_rns() [2/2]	157
11.1.3.250 fconvert_rns() [2/2]	157
11.1.3.251 freduce() [8/11]	157
11.1.3.252 freduce() [9/11]	157
11.1.3.253 finit() [7/8]	158
11.1.3.254 fconvert() [3/3]	158
11.1.3.255 fnegin() [3/4]	159
11.1.3.256 fneg() [3/4]	159
11.1.3.257 fzero() [4/5]	160
11.1.3.258 fiszero() [3/4]	160
11.1.3.259 fequal() [3/4]	160

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

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

11.1.3.344 getSeed()	189
11.2 FFLAS::_frtranspose_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()	205
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	206
11.9 FFLAS::FieldCategories Namespace Reference	206
11.9.1 Detailed Description	207
11.10 FFLAS::MMHelperAlgo Namespace Reference	207
11.11 FFLAS::ModeCategories Namespace Reference	207
11.11.1 Detailed Description	207
11.12 FFLAS::ParSeqHelper Namespace Reference	208
11.12.1 Detailed Description	208
11.13 FFLAS::Protected Namespace Reference	208
11.13.1 Function Documentation	211
11.13.1.1 computeFactorClassic() [1/3]	211
11.13.1.2 computeFactorClassic() [2/3]	211
11.13.1.3 computeFactorClassic() [3/3]	212
11.13.1.4 DotProdBoundClassic()	212
11.13.1.5 TRSMBound() [1/3]	212
11.13.1.6 TRSMBound() [2/3]	212
11.13.1.7 TRSMBound() [3/3]	212
11.13.1.8 fgemm_convert()	213
11.13.1.9 NeedPreAddReduction() [1/2]	213
11.13.1.10 NeedPreAddReduction() [2/2]	213
11.13.1.11 NeedPreSubReduction() [1/2]	213
11.13.1.12 NeedPreSubReduction() [2/2]	214
11.13.1.13 NeedDoublePreAddReduction() [1/2]	214
11.13.1.14 NeedDoublePreAddReduction() [2/2]	214
11.13.1.15 ScalAndReduce() [1/3]	214
11.13.1.16 ScalAndReduce() [2/3]	215
11.13.1.17 fsquareCommon()	215
11.13.1.18 WinogradThreshold() [1/4]	215
11.13.1.19 WinogradThreshold() [2/4]	215

11.13.1.20 WinogradThreshold() [3/4]	215
11.13.1.21 WinogradThreshold() [4/4]	216
11.13.1.22 WinogradSteps()	216
11.13.1.23 DynamicPeeling()	216
11.13.1.24 DynamicPeeling2()	217
11.13.1.25 WinogradCalc()	217
11.13.1.26 fgemv_convert()	217
11.13.1.27 fger_convert()	218
11.13.1.28 fsyrk_convert()	218
11.13.1.29 ScalAndReduce() [3/3]	218
11.13.1.30 NeedPreScalReduction() [1/2]	219
11.13.1.31 NeedPreScalReduction() [2/2]	219
11.13.1.32 NeedPreAxyReduction() [1/2]	219
11.13.1.33 NeedPreAxyReduction() [2/2]	219
11.13.1.34 min_types() [1/7]	219
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]	220
11.13.1.39 min_types() [6/7]	220
11.13.1.40 min_types() [7/7]	220
11.13.1.41 unfit() [1/4]	220
11.13.1.42 unfit() [2/4]	220
11.13.1.43 unfit() [3/4]	221
11.13.1.44 unfit() [4/4]	221
11.13.1.45 igemm_colmajor() [1/2]	221
11.13.1.46 igemm_colmajor() [2/2]	221
11.13.1.47 igemm()	221
11.13.1.48 MatF2MatD_Triangular()	222
11.13.1.49 MatF2MatFI_Triangular()	222
11.14 FFLAS::sell_details Namespace Reference	222
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]	226
11.15.1.3 fspmv_dispatch() [1/2]	226
11.15.1.4 fspmv_dispatch() [2/2]	226
11.15.1.5 fspmv() [1/12]	227
11.15.1.6 fspmv() [2/12]	227
11.15.1.7 fspmv() [3/12]	227
11.15.1.8 fspmv() [4/12]	227
11.15.1.9 fspmv() [5/12]	227

11.15.1.10 fspmv()	[ 6/12]	228
11.15.1.11 fspmv()	[ 7/12]	228
11.15.1.12 fspmv()	[ 8/12]	228
11.15.1.13 fspmv()	[ 9/12]	228
11.15.1.14 fspmm_dispatch()	[ 1/2]	229
11.15.1.15 fspmm_dispatch()	[ 2/2]	229
11.15.1.16 fspmm()	[ 1/9]	229
11.15.1.17 fspmm()	[ 2/9]	229
11.15.1.18 fspmm()	[ 3/9]	230
11.15.1.19 fspmm()	[ 4/9]	230
11.15.1.20 fspmm()	[ 5/9]	230
11.15.1.21 fspmm()	[ 6/9]	231
11.15.1.22 fspmm()	[ 7/9]	231
11.15.1.23 fspmm()	[ 8/9]	231
11.15.1.24 fspmm()	[ 9/9]	231
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]	232
11.15.1.28 pfspmm()	[ 2/9]	232
11.15.1.29 pfspmm()	[ 3/9]	233
11.15.1.30 pfspmm()	[ 4/9]	233
11.15.1.31 pfspmm()	[ 5/9]	233
11.15.1.32 pfspmm()	[ 6/9]	234
11.15.1.33 pfspmm()	[ 7/9]	234
11.15.1.34 pfspmm()	[ 8/9]	234
11.15.1.35 pfspmm()	[ 9/9]	234
11.15.1.36 pfspmv()	[ 1/6]	235
11.15.1.37 pfspmv()	[ 2/6]	235
11.15.1.38 pfspmv()	[ 3/6]	235
11.15.1.39 pfspmv()	[ 4/6]	235
11.15.1.40 pfspmv()	[ 5/6]	235
11.15.1.41 pfspmv()	[ 6/6]	236
11.15.1.42 fspmv()	[10/12]	236
11.15.1.43 fspmv()	[11/12]	236
11.15.1.44 fspmv()	[12/12]	236
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]	245
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]	246



11.16.1.6 fspmm_one() [1/4]	246
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]	247
11.16.1.10 fspmm_mone_simd_aligned() [1/3]	247
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]	248
11.16.1.14 fspmv() [3/21]	248
11.16.1.15 fspmv_one() [1/10]	248
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]	249
11.16.1.19 pfspmm() [1/18]	249
11.16.1.20 pfspmm() [2/18]	249
11.16.1.21 pfspmm() [3/18]	250
11.16.1.22 pfspmm_one() [1/2]	250
11.16.1.23 pfspmm_mone() [1/2]	250
11.16.1.24 pfspmm_one() [2/2]	250
11.16.1.25 pfspmm_mone() [2/2]	251
11.16.1.26 pfspmv() [1/18]	251
11.16.1.27 pfspmv_task()	251
11.16.1.28 pfspmv() [2/18]	251
11.16.1.29 pfspmv() [3/18]	251
11.16.1.30 pfspmv_one() [1/8]	252
11.16.1.31 pfspmv_mone() [1/8]	252
11.16.1.32 pfspmv_one() [2/8]	252
11.16.1.33 pfspmv_mone() [2/8]	252
11.16.1.34 fspmm() [4/15]	252
11.16.1.35 fspmm() [5/15]	253
11.16.1.36 fspmm_simd_aligned() [2/2]	253
11.16.1.37 fspmm_simd_unaligned() [2/2]	253
11.16.1.38 fspmm() [6/15]	253
11.16.1.39 fspmm_one() [2/4]	254
11.16.1.40 fspmm_mone() [2/4]	254
11.16.1.41 fspmm_one_simd_aligned() [2/3]	254
11.16.1.42 fspmm_one_simd_unaligned() [2/3]	254
11.16.1.43 fspmm_mone_simd_aligned() [2/3]	255
11.16.1.44 fspmm_mone_simd_unaligned() [2/3]	255
11.16.1.45 fspmv() [4/21]	255
11.16.1.46 fspmv() [5/21]	255
11.16.1.47 fspmv() [6/21]	255

11.16.1.48 fspmv_one() [3/10]	256
11.16.1.49 fspmv_mone() [3/10]	256
11.16.1.50 fspmv_one() [4/10]	256
11.16.1.51 fspmv_mone() [4/10]	256
11.16.1.52 pfsppmm() [4/18]	256
11.16.1.53 pfsppmm() [5/18]	257
11.16.1.54 pfsppmm() [6/18]	257
11.16.1.55 pfsppmm() [7/18]	257
11.16.1.56 pfsppmm() [8/18]	257
11.16.1.57 pfsppmm() [9/18]	258
11.16.1.58 pfsppmv() [4/18]	258
11.16.1.59 pfsppmv() [5/18]	258
11.16.1.60 pfsppmv() [6/18]	258
11.16.1.61 fspmm() [7/15]	258
11.16.1.62 fspmm() [8/15]	259
11.16.1.63 fspmm() [9/15]	259
11.16.1.64 fspmv() [7/21]	259
11.16.1.65 fspmv() [8/21]	259
11.16.1.66 fspmv() [9/21]	259
11.16.1.67 pfsppmm() [10/18]	260
11.16.1.68 pfsppmm() [11/18]	260
11.16.1.69 pfsppmm() [12/18]	260
11.16.1.70 pfsppmm() [13/18]	260
11.16.1.71 pfsppmm() [14/18]	261
11.16.1.72 pfsppmm() [15/18]	261
11.16.1.73 pfsppmm_zo() [1/2]	261
11.16.1.74 pfsppmm_zo() [2/2]	261
11.16.1.75 pfsppmv() [7/18]	262
11.16.1.76 pfsppmv() [8/18]	262
11.16.1.77 pfsppmv() [9/18]	262
11.16.1.78 pfsppmv_one() [3/8]	262
11.16.1.79 pfsppmv_mone() [3/8]	262
11.16.1.80 pfsppmv_one() [4/8]	263
11.16.1.81 pfsppmv_mone() [4/8]	263
11.16.1.82 fspmm() [10/15]	263
11.16.1.83 fspmm() [11/15]	263
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]	264
11.16.1.87 fspmm_mone() [4/4]	264
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]	265
11.16.1.91 fspmm_mone_simd_aligned() [3/3]	265
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]	266
11.16.1.95 fspmv() [12/21]	266
11.16.1.96 fspmv_one() [5/10]	266
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]	267
11.16.1.100 pfspmv() [10/18]	267
11.16.1.101 pfspmv() [11/18]	267
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]	268
11.16.1.105 pfspmv_one() [6/8]	268
11.16.1.106 pfspmv_mone() [6/8]	268
11.16.1.107 fspmv() [13/21]	269
11.16.1.108 fspmv_simd() [1/4]	269
11.16.1.109 fspmv() [14/21]	269
11.16.1.110 fspmv_simd() [2/4]	269
11.16.1.111 fspmv() [15/21]	269
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]	270
11.16.1.115 fspmv_mone() [8/10]	270
11.16.1.116 fspmv_one_simd() [1/2]	270
11.16.1.117 fspmv_mone_simd() [1/2]	271
11.16.1.118 pfspmm() [16/18]	271
11.16.1.119 pfspmm() [17/18]	271
11.16.1.120 pfspmm() [18/18]	271
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]	272
11.16.1.125 fspmm() [14/15]	272
11.16.1.126 fspmm() [15/15]	273
11.16.1.127 fspmv() [16/21]	273
11.16.1.128 fspmv() [17/21]	273
11.16.1.129 fspmv() [18/21]	273
11.16.1.130 pfspmv() [16/18]	273
11.16.1.131 pfspmv() [17/18]	274

11.16.1.132 pfspmv() [18/18]	274
11.16.1.133 pfspmv_one() [7/8]	274
11.16.1.134 pfspmv_mone() [7/8]	274
11.16.1.135 pfspmv_one() [8/8]	274
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]	275
11.16.1.139 fspmv() [20/21]	275
11.16.1.140 fspmv_simd() [4/4]	275
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]	276
11.16.1.144 fspmv_one_simd() [2/2]	276
11.16.1.145 fspmv_mone_simd() [2/2]	276
11.16.1.146 fspmv_one() [10/10]	277
11.16.1.147 fspmv_mone() [10/10]	277
11.17 FFLAS::StrategyParameter Namespace Reference	277
11.18 FFLAS::StructureHelper Namespace Reference	277
11.18.1 Detailed Description	277
11.19 FFLAS::vectorised Namespace Reference	278
11.19.1 Function Documentation	279
11.19.1.1 VEC_ADD()	279
11.19.1.2 addp()	279
11.19.1.3 VEC_SUB()	280
11.19.1.4 subp()	280
11.19.1.5 add()	280
11.19.1.6 sub()	280
11.19.1.7 axpyp() [1/2]	280
11.19.1.8 axpyp() [2/2]	281
11.19.1.9 reduce() [1/9]	281
11.19.1.10 reduce() [2/9]	281
11.19.1.11 reduce() [3/9]	281
11.19.1.12 reduce() [4/9]	281
11.19.1.13 reduce() [5/9]	281
11.19.1.14 reduce() [6/9]	282
11.19.1.15 reduce() [7/9]	282
11.19.1.16 reduce() [8/9]	282
11.19.1.17 reduce() [9/9]	282
11.19.1.18 modp() [1/2]	282
11.19.1.19 modp() [2/2]	282
11.19.1.20 scalp() [1/3]	283
11.19.1.21 scalp() [2/3]	283

11.19.1.22 scalp() [3/3]	283
11.20 FFLAS::vectorised::unswitch Namespace Reference	283
11.20.1 Function Documentation	284
11.20.1.1 axpyp() [1/2]	284
11.20.1.2 axpyp() [2/2]	284
11.20.1.3 modp() [1/2]	285
11.20.1.4 modp() [2/2]	285
11.20.1.5 scalp() [1/3]	285
11.20.1.6 scalp() [2/3]	285
11.20.1.7 scalp() [3/3]	286
11.21 FFPACK Namespace Reference	286
11.21.1 Detailed Description	302
11.21.2 Typedef Documentation	302
11.21.2.1 Checker_PLUQ	302
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	303
11.21.2.6 ForceCheck_Det	303
11.21.2.7 ForceCheck_invert	303
11.21.2.8 ForceCheck_charpoly	303
11.21.3 Function Documentation	303
11.21.3.1 LAPACKPerm2MathPerm()	303
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()	305
11.21.3.7 fgetrs() [1/4]	306
11.21.3.8 fgetrs() [2/4]	307
11.21.3.9 fgesv() [1/4]	307
11.21.3.10 fgesv() [2/4]	308
11.21.3.11 ftrtri() [1/2]	309
11.21.3.12 trinv_left() [1/2]	309
11.21.3.13 ftrtrm() [1/2]	310
11.21.3.14 ftrstr()	310
11.21.3.15 ftrssyr2k()	311
11.21.3.16 fsytrf() [1/3]	311
11.21.3.17 fsytrf() [2/3]	312
11.21.3.18 fsytrf() [3/3]	312
11.21.3.19 fsytrf_nonunit() [1/3]	312
11.21.3.20 PLUQ() [1/6]	313

11.21.3.21 pPLUQ()	314
11.21.3.22 PLUQ() [2/6]	314
11.21.3.23 PLUQ() [3/6]	314
11.21.3.24 LUdivine() [1/4]	314
11.21.3.25 ColumnEchelonForm() [1/3]	315
11.21.3.26 pColumnEchelonForm()	316
11.21.3.27 ColumnEchelonForm() [2/3]	316
11.21.3.28 RowEchelonForm() [1/3]	316
11.21.3.29 pRowEchelonForm()	317
11.21.3.30 RowEchelonForm() [2/3]	317
11.21.3.31 ReducedColumnEchelonForm() [1/3]	318
11.21.3.32 pReducedColumnEchelonForm()	318
11.21.3.33 ReducedColumnEchelonForm() [2/3]	318
11.21.3.34 ReducedRowEchelonForm() [1/3]	319
11.21.3.35 pReducedRowEchelonForm()	319
11.21.3.36 ReducedRowEchelonForm() [2/3]	320
11.21.3.37 Invert() [1/4]	320
11.21.3.38 Invert() [2/4]	320
11.21.3.39 Invert2() [1/2]	321
11.21.3.40 CharPoly() [1/8]	322
11.21.3.41 CharPoly() [2/8]	322
11.21.3.42 CharPoly() [3/8]	323
11.21.3.43 MinPoly() [1/4]	323
11.21.3.44 MinPoly() [2/4]	324
11.21.3.45 MatVecMinPoly() [1/2]	324
11.21.3.46 Rank() [1/3]	325
11.21.3.47 pRank()	325
11.21.3.48 Rank() [2/3]	325
11.21.3.49 IsSingular() [1/2]	326
11.21.3.50 Det() [1/6]	326
11.21.3.51 pDet()	327
11.21.3.52 Det() [2/6]	327
11.21.3.53 Solve() [1/3]	327
11.21.3.54 Solve() [2/3]	328
11.21.3.55 pSolve()	328
11.21.3.56 RandomNullSpaceVector() [1/3]	328
11.21.3.57 NullSpaceBasis() [1/2]	329
11.21.3.58 RowRankProfile() [1/3]	329
11.21.3.59 pRowRankProfile()	330
11.21.3.60 RowRankProfile() [2/3]	330
11.21.3.61 ColumnRankProfile() [1/3]	330
11.21.3.62 pColumnRankProfile()	331

11.21.3.63 ColumnRankProfile() [2/3]	331
11.21.3.64 RankProfileFromLU()	331
11.21.3.65 LeadingSubmatrixRankProfiles()	332
11.21.3.66 RowRankProfileSubmatrixIndices() [1/2]	332
11.21.3.67 ColRankProfileSubmatrixIndices() [1/2]	333
11.21.3.68 RowRankProfileSubmatrix() [1/2]	334
11.21.3.69 ColRankProfileSubmatrix() [1/2]	334
11.21.3.70 getTriangular() [1/2]	335
11.21.3.71 getTriangular() [2/2]	335
11.21.3.72 getEchelonForm() [1/2]	336
11.21.3.73 getEchelonForm() [2/2]	337
11.21.3.74 getEchelonTransform()	337
11.21.3.75 getReducedEchelonForm() [1/2]	338
11.21.3.76 getReducedEchelonForm() [2/2]	339
11.21.3.77 getReducedEchelonTransform()	339
11.21.3.78 PLUQtoEchelonPermutation()	340
11.21.3.79 LTBruhatGen()	340
11.21.3.80 getLTBruhatGen() [1/2]	341
11.21.3.81 getLTBruhatGen() [2/2]	341
11.21.3.82 LTQSorder()	342
11.21.3.83 CompressToBlockBiDiagonal()	342
11.21.3.84 ExpandBlockBiDiagonalToBruhat()	343
11.21.3.85 Bruhat2EchelonPermutation()	344
11.21.3.86 TInverter() [1/2]	344
11.21.3.87 ComputeRPermutation() [1/2]	344
11.21.3.88 productBruhatxTS() [1/2]	344
11.21.3.89 LQUPtoInverseOfFullRankMinor() [1/2]	345
11.21.3.90 RandomNullSpaceVector() [2/3]	346
11.21.3.91 solveLB() [1/2]	346
11.21.3.92 solveLB2() [1/2]	346
11.21.3.93 TInverter() [2/2]	347
11.21.3.94 ComputeRPermutation() [2/2]	347
11.21.3.95 expandLCRE()	347
11.21.3.96 productBruhatxTS() [2/2]	348
11.21.3.97 Danilevski()	349
11.21.3.98 buildMatrix()	349
11.21.3.99 CharPoly() [4/8]	349
11.21.3.100 CharPoly() [5/8]	350
11.21.3.101 Det() [3/6]	350
11.21.3.102 Det() [4/6]	350
11.21.3.103 fsytrf_BC_Crout()	350
11.21.3.104 fsytrf_BC_RL()	351

11.21.3.105 fsytrf_UP_RPM_BC_RL()	351
11.21.3.106 fsytrf_LOW_RPM_BC_Crout()	351
11.21.3.107 fsytrf_UP_RPM_BC_Crout()	351
11.21.3.108 fsytrf_UP_RPM()	352
11.21.3.109 fsytrf_nonunit() [2/3]	352
11.21.3.110 fsytrf_nonunit() [3/3]	352
11.21.3.111 fsytrf_RPM()	352
11.21.3.112 getTridiagonal()	353
11.21.3.113 LUdivine_gauss() [1/2]	353
11.21.3.114 LUdivine_small() [1/2]	353
11.21.3.115 LUdivine() [2/4]	353
11.21.3.116 LUdivine() [3/4]	354
11.21.3.117 MonotonicCompress()	354
11.21.3.118 MonotonicCompressMorePivots()	354
11.21.3.119 MonotonicCompressCycles()	355
11.21.3.120 MonotonicExpand()	355
11.21.3.121 applyP_block()	355
11.21.3.122 doApplyS()	355
11.21.3.123 MatrixApplyS() [1/3]	356
11.21.3.124 MatrixApplyS() [2/3]	356
11.21.3.125 MatrixApplyS() [3/3]	356
11.21.3.126 PermApplyS()	356
11.21.3.127 doApplyT()	357
11.21.3.128 MatrixApplyT() [1/3]	357
11.21.3.129 MatrixApplyT() [2/3]	357
11.21.3.130 MatrixApplyT() [3/3]	357
11.21.3.131 PermApplyT()	358
11.21.3.132 composePermutationsLLL()	358
11.21.3.133 composePermutationsLLM()	358
11.21.3.134 composePermutationsMLM()	359
11.21.3.135 cyclic_shift_mathPerm()	359
11.21.3.136 cyclic_shift_row_col() [1/2]	359
11.21.3.137 cyclic_shift_row() [1/3]	359
11.21.3.138 cyclic_shift_row() [2/3]	360
11.21.3.139 cyclic_shift_col() [1/3]	360
11.21.3.140 cyclic_shift_col() [2/3]	360
11.21.3.141 PLUQ_basecaseV3()	360
11.21.3.142 PLUQ_basecaseV2()	360
11.21.3.143 PLUQ_basecaseCrout()	361
11.21.3.144 _PLUQ()	361
11.21.3.145 PLUQ() [4/6]	361
11.21.3.146 threads_fgemm()	361



11.21.3.147 threads_frsm()	362
11.21.3.148 PLUQ() [5/6]	362
11.21.3.149 fflas_const_cast() [1/3]	362
11.21.3.150 fflas_const_cast() [2/3]	362
11.21.3.151 cyclic_shift_row_col() [2/2]	362
11.21.3.152 cyclic_shift_row() [3/3]	363
11.21.3.153 cyclic_shift_col() [3/3]	363
11.21.3.154 applyP() [4/4]	363
11.21.3.155 fgetrs() [3/4]	363
11.21.3.156 fgetrs() [4/4]	364
11.21.3.157 fgesv() [3/4]	364
11.21.3.158 fgesv() [4/4]	364
11.21.3.159 ftrtri() [2/2]	364
11.21.3.160 trinv_left() [2/2]	365
11.21.3.161 ftrtm() [2/2]	365
11.21.3.162 PLUQ() [6/6]	365
11.21.3.163 LUdivine() [4/4]	365
11.21.3.164 LUdivine_small() [2/2]	366
11.21.3.165 LUdivine_gauss() [2/2]	366
11.21.3.166 RowEchelonForm() [3/3]	366
11.21.3.167 ReducedRowEchelonForm() [3/3]	366
11.21.3.168 ColumnEchelonForm() [3/3]	367
11.21.3.169 ReducedColumnEchelonForm() [3/3]	367
11.21.3.170 Invert() [3/4]	367
11.21.3.171 Invert() [4/4]	367
11.21.3.172 Invert2() [2/2]	367
11.21.3.173 CharPoly() [6/8]	368
11.21.3.174 CharPoly() [7/8]	368
11.21.3.175 CharPoly() [8/8]	368
11.21.3.176 MinPoly() [3/4]	368
11.21.3.177 MinPoly() [4/4]	368
11.21.3.178 MatVecMinPoly() [2/2]	369
11.21.3.179 KrylovElim()	369
11.21.3.180 SpecRankProfile()	369
11.21.3.181 Rank() [3/3]	369
11.21.3.182 IsSingular() [2/2]	369
11.21.3.183 Det() [5/6]	370
11.21.3.184 Det() [6/6]	370
11.21.3.185 Solve() [3/3]	370
11.21.3.186 solveLB() [2/2]	370
11.21.3.187 solveLB2() [2/2]	371
11.21.3.188 RandomNullSpaceVector() [3/3]	371

11.21.3.189 NullSpaceBasis() [2/2] . . . . .	371
11.21.3.190 RowRankProfile() [3/3] . . . . .	371
11.21.3.191 ColumnRankProfile() [3/3] . . . . .	372
11.21.3.192 RowRankProfileSubmatrixIndices() [2/2] . . . . .	372
11.21.3.193 ColRankProfileSubmatrixIndices() [2/2] . . . . .	372
11.21.3.194 RowRankProfileSubmatrix() [2/2] . . . . .	372
11.21.3.195 ColRankProfileSubmatrix() [2/2] . . . . .	372
11.21.3.196 getTriangular< FFLAS_FIELD< FFLAS_ELT > >() [1/2] . . . . .	373
11.21.3.197 getTriangular< FFLAS_FIELD< FFLAS_ELT > >() [2/2] . . . . .	373
11.21.3.198 getEchelonForm< FFLAS_FIELD< FFLAS_ELT > >() [1/2] . . . . .	373
11.21.3.199 getEchelonForm< FFLAS_FIELD< FFLAS_ELT > >() [2/2] . . . . .	373
11.21.3.200 getEchelonTransform< FFLAS_FIELD< FFLAS_ELT > >() . . . . .	374
11.21.3.201 getReducedEchelonForm< FFLAS_FIELD< FFLAS_ELT > >() [1/2] . . . . .	374
11.21.3.202 getReducedEchelonForm< FFLAS_FIELD< FFLAS_ELT > >() [2/2] . . . . .	374
11.21.3.203 getReducedEchelonTransform< FFLAS_FIELD< FFLAS_ELT > >() . . . . .	374
11.21.3.204 LQUPtoInverseOfFullRankMinor() [2/2] . . . . .	375
11.21.3.205 fflas_const_cast() [3/3] . . . . .	375
11.21.3.206 failure() . . . . .	375
11.21.3.207 isOdd() [1/3] . . . . .	375
11.21.3.208 isOdd() [2/3] . . . . .	375
11.21.3.209 isOdd() [3/3] . . . . .	375
11.21.3.210 NonZeroRandomMatrix() [1/2] . . . . .	376
11.21.3.211 NonZeroRandomMatrix() [2/2] . . . . .	376
11.21.3.212 RandomMatrix() [1/2] . . . . .	377
11.21.3.213 RandomMatrix() [2/2] . . . . .	377
11.21.3.214 RandomTriangularMatrix() [1/2] . . . . .	378
11.21.3.215 RandomTriangularMatrix() [2/2] . . . . .	378
11.21.3.216 RandInt() . . . . .	379
11.21.3.217 RandomSymmetricMatrix() . . . . .	379
11.21.3.218 RandomMatrixWithRank() [1/2] . . . . .	380
11.21.3.219 RandomMatrixWithRank() [2/2] . . . . .	380
11.21.3.220 RandomIndexSubset() . . . . .	381
11.21.3.221 RandomPermutation() . . . . .	381
11.21.3.222 RandomRankProfileMatrix() . . . . .	381
11.21.3.223 swapval() . . . . .	382
11.21.3.224 RandomSymmetricRankProfileMatrix() . . . . .	382
11.21.3.225 RandomLTQSRankProfileMatrix() . . . . .	382
11.21.3.226 RandomMatrixWithRankandRPM() [1/2] . . . . .	382
11.21.3.227 RandomMatrixWithRankandRPM() [2/2] . . . . .	383
11.21.3.228 RandomSymmetricMatrixWithRankandRPM() [1/2] . . . . .	384
11.21.3.229 RandomSymmetricMatrixWithRankandRPM() [2/2] . . . . .	384
11.21.3.230 RandomMatrixWithRankandRandomRPM() [1/2] . . . . .	385

11.21.3.231 RandomMatrixWithRankandRandomRPM() [2/2]	385
11.21.3.232 RandomSymmetricMatrixWithRankandRandomRPM() [1/2]	386
11.21.3.233 RandomSymmetricMatrixWithRankandRandomRPM() [2/2]	386
11.21.3.234 RandomMatrixWithDet() [1/2]	387
11.21.3.235 RandomMatrixWithDet() [2/2]	387
11.21.3.236 RandomLTQSMatrixWithRankandQSorder()	388
11.21.3.237 chooseField()	388
11.21.3.238 chooseField< Givaro::ZRing< int32_t > >()	388
11.21.3.239 chooseField< Givaro::ZRing< int64_t > >()	388
11.21.3.240 chooseField< Givaro::ZRing< float > >()	389
11.21.3.241 chooseField< Givaro::ZRing< double > >()	389
11.22 FFPACK::Protected Namespace Reference	389
11.22.1 Function Documentation	390
11.22.1.1 LUdivine_construct() [1/2]	390
11.22.1.2 GaussJordan()	391
11.22.1.3 KellerGehrig()	392
11.22.1.4 KGFast()	392
11.22.1.5 KGFast_generalized()	392
11.22.1.6 fgemv_kgf()	392
11.22.1.7 LUKrylov()	392
11.22.1.8 Danilevski()	393
11.22.1.9 RandomKrylovPrecond()	393
11.22.1.10 ArithProg()	393
11.22.1.11 LUKrylov_KGFast()	393
11.22.1.12 MatVecMinPoly()	394
11.22.1.13 Hybrid_KGF_LUK_MinPoly()	394
11.22.1.14 updateD()	394
11.22.1.15 newD()	394
11.22.1.16 CompressRows()	394
11.22.1.17 CompressRowsQK()	395
11.22.1.18 DeCompressRows()	395
11.22.1.19 DeCompressRowsQK()	395
11.22.1.20 CompressRowsQA()	395
11.22.1.21 DeCompressRowsQA()	396
11.22.1.22 LUdivine_construct() [2/2]	396
11.23 Givaro Namespace Reference	396
11.24 MKL_CONFIG Namespace Reference	396
11.25 Reclnt Namespace Reference	396
<b>12 Data Structure Documentation</b>	<b>397</b>
12.1 AlgoChooser< ModeT, ParSeq > Struct Template Reference	397
12.1.1 Member Typedef Documentation	397

12.1.1.1 value	397
12.2 ALL< v > Struct Template Reference	397
12.3 ArbitraryPrecIntTag Struct Reference	397
12.3.1 Detailed Description	397
12.4 AreEqual< X, Y > Class Template Reference	397
12.4.1 Field Documentation	398
12.4.1.1 value	398
12.5 Argument Struct Reference	398
12.5.1 Field Documentation	398
12.5.1.1 c	398
12.5.1.2 example	398
12.5.1.3 helpString	398
12.5.1.4 type	398
12.5.1.5 data	398
12.6 array< T > Class Template Reference	398
12.6.1 Detailed Description	399
12.6.2 Field Documentation	399
12.6.2.1 elements	399
12.7 associatedDelayedField< Field > Struct Template Reference	399
12.7.1 Member Typedef Documentation	399
12.7.1.1 field	399
12.7.1.2 type	399
12.8 Auto Struct Reference	399
12.9 Bench< Elt > Class Template Reference	399
12.9.1 Member Typedef Documentation	400
12.9.1.1 Field	400
12.9.1.2 Elt_ptr	400
12.9.1.3 Residu	400
12.9.1.4 enable_if_t	400
12.9.1.5 is_same_element	400
12.9.1.6 enable_if_no_simd_t	401
12.9.1.7 enable_if_simd128_t	401
12.9.1.8 enable_if_simd256_t	401
12.9.1.9 enable_if_simd512_t	401
12.9.2 Constructor & Destructor Documentation	401
12.9.2.1 Bench()	401
12.9.3 Member Function Documentation	401
12.9.3.1 cardinality() [1/2]	401
12.9.3.2 cardinality() [2/2]	401
12.9.3.3 doBenchs()	401
12.9.3.4 run()	401
12.9.4 Field Documentation	402

12.9.4.1 F	402
12.9.4.2 m	402
12.9.4.3 n	402
12.9.4.4 iters	402
12.9.4.5 inplace	402
12.10 Bini Struct Reference	402
12.11 Block Struct Reference	402
12.12 BlockTransposeSIMD< Field, Simd, > Struct Template Reference	402
12.12.1 Member Function Documentation	403
12.12.1.1 size()	403
12.12.1.2 info()	403
12.12.1.3 transpose() [1/5]	403
12.12.1.4 transpose() [2/5]	403
12.12.1.5 transpose() [3/5]	403
12.12.1.6 transpose() [4/5]	404
12.12.1.7 transpose() [5/5]	404
12.13 callLUdivine_small< Element > Class Template Reference	404
12.13.1 Member Function Documentation	404
12.13.1.1 operator()	404
12.14 CharpolyFailed Class Reference	404
12.15 Checker_Empty< Field > Struct Template Reference	405
12.15.1 Constructor & Destructor Documentation	405
12.15.1.1 Checker_Empty()	405
12.15.2 Member Function Documentation	405
12.15.2.1 check()	405
12.16 CheckerImplem_charpoly< Field, Polynomial > Class Template Reference	405
12.16.1 Constructor & Destructor Documentation	405
12.16.1.1 CheckerImplem_charpoly() [1/2]	405
12.16.1.2 CheckerImplem_charpoly() [2/2]	406
12.16.1.3 ~CheckerImplem_charpoly()	406
12.16.2 Member Function Documentation	406
12.16.2.1 check()	406
12.17 CheckerImplem_Det< Field > Class Template Reference	406
12.17.1 Constructor & Destructor Documentation	406
12.17.1.1 CheckerImplem_Det() [1/2]	406
12.17.1.2 CheckerImplem_Det() [2/2]	406
12.17.1.3 ~CheckerImplem_Det()	407
12.17.2 Member Function Documentation	407
12.17.2.1 check()	407
12.18 CheckerImplem_fgemm< Field > Class Template Reference	407
12.18.1 Constructor & Destructor Documentation	407
12.18.1.1 CheckerImplem_fgemm() [1/2]	407

12.18.1.2 CheckerImplem_fgemm() [2/2]	408
12.18.1.3 ~CheckerImplem_fgemm()	408
12.18.2 Member Function Documentation	408
12.18.2.1 check()	408
12.19 CheckerImplem_ftsm< Field > Class Template Reference	408
12.19.1 Constructor & Destructor Documentation	408
12.19.1.1 CheckerImplem_ftsm() [1/2]	408
12.19.1.2 CheckerImplem_ftsm() [2/2]	409
12.19.1.3 ~CheckerImplem_ftsm()	409
12.19.2 Member Function Documentation	409
12.19.2.1 check()	409
12.20 CheckerImplem_invert< Field > Class Template Reference	409
12.20.1 Constructor & Destructor Documentation	409
12.20.1.1 CheckerImplem_invert() [1/2]	409
12.20.1.2 CheckerImplem_invert() [2/2]	410
12.20.1.3 ~CheckerImplem_invert()	410
12.20.2 Member Function Documentation	410
12.20.2.1 check()	410
12.21 CheckerImplem_PLUQ< Field > Class Template Reference	410
12.21.1 Constructor & Destructor Documentation	410
12.21.1.1 CheckerImplem_PLUQ() [1/2]	410
12.21.1.2 CheckerImplem_PLUQ() [2/2]	411
12.21.1.3 ~CheckerImplem_PLUQ()	411
12.21.2 Member Function Documentation	411
12.21.2.1 check()	411
12.22 Classic Struct Reference	411
12.23 Column Struct Reference	411
12.24 CompactElement< Element > Struct Template Reference	411
12.24.1 Member Typedef Documentation	412
12.24.1.1 type	412
12.25 compatible_data_type< Field > Struct Template Reference	412
12.25.1 Field Documentation	412
12.25.1.1 value	412
12.26 Compose< H1, H2 > Struct Template Reference	412
12.26.1 Constructor & Destructor Documentation	412
12.26.1.1 Compose() [1/5]	412
12.26.1.2 Compose() [2/5]	412
12.26.1.3 Compose() [3/5]	412
12.26.1.4 Compose() [4/5]	413
12.26.1.5 Compose() [5/5]	413
12.26.2 Member Function Documentation	413
12.26.2.1 first_component()	413

12.26.2.2 second_component()	413
12.26.3 Friends And Related Symbol Documentation	413
12.26.3.1 operator<<	413
12.27 array< T >::const_iterator Class Reference	413
12.28 string::const_iterator Class Reference	413
12.29 vector< T >::const_iterator Class Reference	413
12.30 array< T >::const_reverse_iterator Class Reference	413
12.31 string::const_reverse_iterator Class Reference	413
12.32 vector< T >::const_reverse_iterator Class Reference	414
12.33 ConvertTo< T > Struct Template Reference	414
12.33.1 Detailed Description	414
12.34 Coo< ValT, IdxT > Struct Template Reference	414
12.34.1 Member Typedef Documentation	414
12.34.1.1 Self	414
12.34.2 Constructor & Destructor Documentation	415
12.34.2.1 Coo() [1/4]	415
12.34.2.2 Coo() [2/4]	415
12.34.2.3 Coo() [3/4]	415
12.34.2.4 Coo() [4/4]	415
12.34.3 Member Function Documentation	415
12.34.3.1 operator=() [1/2]	415
12.34.3.2 operator=() [2/2]	415
12.34.4 Field Documentation	415
12.34.4.1 val	415
12.34.4.2 row	415
12.34.4.3 col	415
12.35 Coo< Field > Struct Template Reference	416
12.35.1 Constructor & Destructor Documentation	416
12.35.1.1 Coo() [1/4]	416
12.35.1.2 Coo() [2/4]	416
12.35.1.3 Coo() [3/4]	416
12.35.1.4 Coo() [4/4]	416
12.35.2 Member Function Documentation	416
12.35.2.1 operator=() [1/2]	416
12.35.2.2 operator=() [2/2]	416
12.35.3 Field Documentation	417
12.35.3.1 val	417
12.35.3.2 col	417
12.35.3.3 row	417
12.35.3.4 deleted	417
12.36 Coo< ValT, IdxT > Struct Template Reference	417
12.36.1 Member Typedef Documentation	417

12.36.1.1 Self	417
12.36.2 Constructor & Destructor Documentation	417
12.36.2.1 Coo() [1/4]	417
12.36.2.2 Coo() [2/4]	418
12.36.2.3 Coo() [3/4]	418
12.36.2.4 Coo() [4/4]	418
12.36.3 Member Function Documentation	418
12.36.3.1 operator=() [1/2]	418
12.36.3.2 operator=() [2/2]	418
12.36.4 Field Documentation	418
12.36.4.1 val	418
12.36.4.2 row	418
12.36.4.3 col	418
12.37 CooMat< Field > Struct Template Reference	418
12.37.1 Field Documentation	419
12.37.1.1 _coo16	419
12.37.1.2 _coo32	419
12.37.1.3 _coo64	419
12.37.1.4 _coo16_zo	419
12.37.1.5 _coo32_zo	419
12.37.1.6 _coo64_zo	419
12.38 count_nonconst_lvalue_reference< T > Struct Template Reference	419
12.39 CsrMat< Field > Struct Template Reference	419
12.39.1 Field Documentation	420
12.39.1.1 _csr16	420
12.39.1.2 _csr32	420
12.39.1.3 _csr64	420
12.39.1.4 _csr16_zo	420
12.39.1.5 _csr32_zo	420
12.39.1.6 _csr64_zo	420
12.40 DefaultBoundedTag Struct Reference	420
12.40.1 Detailed Description	420
12.41 DefaultTag Struct Reference	420
12.41.1 Detailed Description	421
12.42 DelayedTag Struct Reference	421
12.42.1 Detailed Description	421
12.43 DivideAndConquer Struct Reference	421
12.44 ElementTraits< Element > Struct Template Reference	421
12.44.1 Detailed Description	421
12.44.2 Member Typedef Documentation	421
12.44.2.1 value	421
12.45 EllMat< Field > Struct Template Reference	421



12.45.1 Field Documentation	422
12.45.1.1 _ell16	422
12.45.1.2 _ell32	422
12.45.1.3 _ell64	422
12.45.1.4 _ell16_zo	422
12.45.1.5 _ell32_zo	422
12.45.1.6 _ell64_zo	422
12.46 Failure Class Reference	422
12.46.1 Detailed Description	423
12.46.2 Constructor & Destructor Documentation	423
12.46.2.1 Failure()	423
12.46.3 Member Function Documentation	423
12.46.3.1 operator>() [1/2]	423
12.46.3.2 operator>() [2/2]	423
12.46.3.3 setErrorStream()	423
12.46.3.4 print()	423
12.46.4 Field Documentation	424
12.46.4.1 _errorStream	424
12.47 FailureCharpolyCheck Class Reference	424
12.48 FailureDetCheck Class Reference	424
12.49 FailureFgemmCheck Class Reference	424
12.50 FailureInvertCheck Class Reference	424
12.51 FailurePLUQCheck Class Reference	424
12.52 FailureTrsmCheck Class Reference	424
12.53 FieldSimd<_Field> Class Template Reference	425
12.53.1 Member Typedef Documentation	426
12.53.1.1 Field	426
12.53.1.2 Element	426
12.53.1.3 simd	426
12.53.1.4 vect_t	426
12.53.1.5 scalar_t	426
12.53.2 Constructor & Destructor Documentation	426
12.53.2.1 FieldSimd() [1/3]	426
12.53.2.2 FieldSimd() [2/3]	426
12.53.2.3 FieldSimd() [3/3]	426
12.53.3 Member Function Documentation	426
12.53.3.1 operator=() [1/2]	426
12.53.3.2 operator=() [2/2]	426
12.53.3.3 init() [1/2]	427
12.53.3.4 init() [2/2]	427
12.53.3.5 add() [1/2]	427
12.53.3.6 add() [2/2]	427

12.53.3.7 addin()	427
12.53.3.8 add_r() [1/2]	427
12.53.3.9 add_r() [2/2]	427
12.53.3.10 addin_r()	427
12.53.3.11 sub() [1/2]	428
12.53.3.12 sub() [2/2]	428
12.53.3.13 subin()	428
12.53.3.14 sub_r() [1/2]	428
12.53.3.15 sub_r() [2/2]	428
12.53.3.16 subin_r()	428
12.53.3.17 zero() [1/2]	428
12.53.3.18 zero() [2/2]	428
12.53.3.19 mod()	428
12.53.3.20 mul() [1/2]	429
12.53.3.21 mul() [2/2]	429
12.53.3.22 mulin()	429
12.53.3.23 mul_r() [1/2]	429
12.53.3.24 mul_r() [2/2]	429
12.53.3.25 axpy() [1/2]	429
12.53.3.26 axpy() [2/2]	429
12.53.3.27 axpyin()	429
12.53.3.28 axpy_r() [1/2]	430
12.53.3.29 axpy_r() [2/2]	430
12.53.3.30 axpyin_r()	430
12.53.3.31 maxpy() [1/2]	430
12.53.3.32 maxpy() [2/2]	430
12.53.3.33 maxpyin()	430
12.53.4 Field Documentation	430
12.53.4.1 vect_size	430
12.53.4.2 alignment	431
12.54 FieldTraits< Field > Struct Template Reference	431
12.54.1 Detailed Description	431
12.54.2 Member Typedef Documentation	431
12.54.2.1 category	431
12.54.3 Field Documentation	431
12.54.3.1 balanced	431
12.55 Fixed Struct Reference	431
12.56 FixedPreIntTag Struct Reference	431
12.56.1 Detailed Description	432
12.57 ForStrategy1D< blocksize_t, Cut, Param > Struct Template Reference	432
12.57.1 Constructor & Destructor Documentation	432
12.57.1.1 ForStrategy1D() [1/2]	432

12.57.1.2 ForStrategy1D() [2/2]	432
12.57.2 Member Function Documentation	432
12.57.2.1 build()	432
12.57.2.2 initialize()	433
12.57.2.3 isTerminated()	433
12.57.2.4 begin()	433
12.57.2.5 end()	433
12.57.2.6 numblocks()	433
12.57.2.7 blockindex()	433
12.57.2.8 operator++()	433
12.57.3 Field Documentation	433
12.57.3.1 ibeg	433
12.57.3.2 iend	433
12.57.3.3 current	434
12.57.3.4 firstBlockSize	434
12.57.3.5 lastBlockSize	434
12.57.3.6 changeBS	434
12.57.3.7 numBlock	434
12.58 ForStrategy2D< blocksize_t, Cut, Param > Struct Template Reference	434
12.58.1 Constructor & Destructor Documentation	435
12.58.1.1 ForStrategy2D()	435
12.58.2 Member Function Documentation	435
12.58.2.1 initialize()	435
12.58.2.2 isTerminated()	435
12.58.2.3 ibegin()	435
12.58.2.4 jbegin()	435
12.58.2.5 iend()	435
12.58.2.6 jend()	436
12.58.2.7 operator++()	436
12.58.2.8 rownumblocks()	436
12.58.2.9 colnumblocks()	436
12.58.2.10 blockindex()	436
12.58.2.11 rowblockindex()	436
12.58.2.12 colblockindex()	436
12.58.3 Friends And Related Symbol Documentation	436
12.58.3.1 operator<<	436
12.58.4 Field Documentation	436
12.58.4.1 _ibeg	436
12.58.4.2 _iend	437
12.58.4.3 _jbeg	437
12.58.4.4 _jend	437
12.58.4.5 rowBlockSize	437

12.58.4.6 colBlockSize . . . . .	437
12.58.4.7 current . . . . .	437
12.58.4.8 lastRBS . . . . .	437
12.58.4.9 lastCBS . . . . .	437
12.58.4.10 changeRBS . . . . .	437
12.58.4.11 changeCBS . . . . .	437
12.58.4.12 numRowsBlock . . . . .	438
12.58.4.13 numColBlock . . . . .	438
12.58.4.14 BLOCKS . . . . .	438
12.59 ftrmmLeftLowerNoTransNonUnit< Element > Class Template Reference . . . . .	438
12.60 ftrmmLeftLowerNoTransUnit< Element > Class Template Reference . . . . .	438
12.61 ftrmmLeftLowerTransNonUnit< Element > Class Template Reference . . . . .	438
12.62 ftrmmLeftLowerTransUnit< Element > Class Template Reference . . . . .	438
12.63 ftrmmLeftUpperNoTransNonUnit< Element > Class Template Reference . . . . .	438
12.64 ftrmmLeftUpperNoTransUnit< Element > Class Template Reference . . . . .	439
12.65 ftrmmLeftUpperTransNonUnit< Element > Class Template Reference . . . . .	439
12.66 ftrmmLeftUpperTransUnit< Element > Class Template Reference . . . . .	439
12.67 ftrmmRightLowerNoTransNonUnit< Element > Class Template Reference . . . . .	439
12.68 ftrmmRightLowerNoTransUnit< Element > Class Template Reference . . . . .	439
12.69 ftrmmRightLowerTransNonUnit< Element > Class Template Reference . . . . .	439
12.70 ftrmmRightLowerTransUnit< Element > Class Template Reference . . . . .	439
12.71 ftrmmRightUpperNoTransNonUnit< Element > Class Template Reference . . . . .	439
12.72 ftrmmRightUpperNoTransUnit< Element > Class Template Reference . . . . .	440
12.73 ftrmmRightUpperTransNonUnit< Element > Class Template Reference . . . . .	440
12.74 ftrmmRightUpperTransUnit< Element > Class Template Reference . . . . .	440
12.75 ftrsmLeftLowerNoTransNonUnit< Element > Class Template Reference . . . . .	440
12.76 ftrsmLeftLowerNoTransUnit< Element > Class Template Reference . . . . .	440
12.77 ftrsmLeftLowerTransNonUnit< Element > Class Template Reference . . . . .	440
12.78 ftrsmLeftLowerTransUnit< Element > Class Template Reference . . . . .	440
12.79 ftrsmLeftUpperNoTransNonUnit< Element > Class Template Reference . . . . .	440
12.79.1 Detailed Description . . . . .	441
12.80 ftrsmLeftUpperNoTransUnit< Element > Class Template Reference . . . . .	441
12.81 ftrsmLeftUpperTransNonUnit< Element > Class Template Reference . . . . .	441
12.82 ftrsmLeftUpperTransUnit< Element > Class Template Reference . . . . .	441
12.83 ftrsmRightLowerNoTransNonUnit< Element > Class Template Reference . . . . .	441
12.84 ftrsmRightLowerNoTransUnit< Element > Class Template Reference . . . . .	441
12.85 ftrsmRightLowerTransNonUnit< Element > Class Template Reference . . . . .	442
12.86 ftrsmRightLowerTransUnit< Element > Class Template Reference . . . . .	442
12.87 ftrsmRightUpperNoTransNonUnit< Element > Class Template Reference . . . . .	442
12.88 ftrsmRightUpperNoTransUnit< Element > Class Template Reference . . . . .	442
12.89 ftrsmRightUpperTransNonUnit< Element > Class Template Reference . . . . .	442
12.90 ftrsmRightUpperTransUnit< Element > Class Template Reference . . . . .	442

12.91 GenericTag Struct Reference . . . . .	442
12.91.1 Detailed Description . . . . .	442
12.92 GenericTag Struct Reference . . . . .	443
12.92.1 Detailed Description . . . . .	443
12.93 Grain Struct Reference . . . . .	443
12.94 has_minus_eq_impl< C > Struct Template Reference . . . . .	443
12.94.1 Field Documentation . . . . .	443
12.94.1.1 value . . . . .	443
12.95 has_minus_impl< C > Struct Template Reference . . . . .	443
12.95.1 Field Documentation . . . . .	443
12.95.1.1 value . . . . .	443
12.96 has_mul_eq_impl< C > Struct Template Reference . . . . .	443
12.96.1 Field Documentation . . . . .	444
12.96.1.1 value . . . . .	444
12.97 has_mul_impl< C > Struct Template Reference . . . . .	444
12.97.1 Field Documentation . . . . .	444
12.97.1.1 value . . . . .	444
12.98 has_operation< T > Struct Template Reference . . . . .	444
12.98.1 Field Documentation . . . . .	444
12.98.1.1 value . . . . .	444
12.99 has_plus_eq_impl< C > Struct Template Reference . . . . .	444
12.99.1 Field Documentation . . . . .	445
12.99.1.1 value . . . . .	445
12.100 has_plus_impl< C > Struct Template Reference . . . . .	445
12.100.1 Field Documentation . . . . .	445
12.100.1.1 value . . . . .	445
12.101 HelperFlag Struct Reference . . . . .	445
12.101.1 Field Documentation . . . . .	445
12.101.1.1 none . . . . .	445
12.101.1.2 coo . . . . .	445
12.101.1.3 csr . . . . .	445
12.101.1.4 ell . . . . .	445
12.101.1.5 aut . . . . .	446
12.101.1.6 pm1 . . . . .	446
12.102 HelperMod< Field, ElementTraits > Struct Template Reference . . . . .	446
12.103 Hybrid Struct Reference . . . . .	446
12.104 Info Struct Reference . . . . .	446
12.104.1 Constructor & Destructor Documentation . . . . .	446
12.104.1.1 Info() [1/4] . . . . .	446
12.104.1.2 Info() [2/4] . . . . .	446
12.104.1.3 Info() [3/4] . . . . .	446
12.104.1.4 Info() [4/4] . . . . .	447

12.104.2 Member Function Documentation	447
12.104.2.1 operator=() [1/2]	447
12.104.2.2 operator=() [2/2]	447
12.104.3 Field Documentation	447
12.104.3.1 size	447
12.104.3.2 perm	447
12.104.3.3 begin	447
12.105 Info Struct Reference	447
12.105.1 Constructor & Destructor Documentation	447
12.105.1.1 Info() [1/4]	447
12.105.1.2 Info() [2/4]	448
12.105.1.3 Info() [3/4]	448
12.105.1.4 Info() [4/4]	448
12.105.2 Member Function Documentation	448
12.105.2.1 operator=() [1/2]	448
12.105.2.2 operator=() [2/2]	448
12.105.3 Field Documentation	448
12.105.3.1 size	448
12.105.3.2 perm	448
12.105.3.3 begin	448
12.106 is_all_same< Args > Struct Template Reference	448
12.107 is_simd< T > Struct Template Reference	448
12.107.1 Member Typedef Documentation	449
12.107.1.1 type	449
12.107.2 Field Documentation	449
12.107.2.1 value	449
12.108 isSparseMatrix< Field, M > Struct Template Reference	449
12.109 isSparseMatrixMKLFormat< F, M > Struct Template Reference	449
12.110 isSparseMatrixSimdFormat< F, M > Struct Template Reference	449
12.111 isZOSparseMatrix< F, M > Struct Template Reference	450
12.112 Iterative Struct Reference	450
12.113 array< T >::iterator Class Reference	450
12.114 string::iterator Class Reference	450
12.115 vector< T >::iterator Class Reference	450
12.116 LazyTag Struct Reference	450
12.116.1 Detailed Description	450
12.117 limits< T > Struct Template Reference	450
12.118 MachineFloatTag Struct Reference	451
12.118.1 Detailed Description	451
12.119 MachineIntTag Struct Reference	451
12.119.1 Detailed Description	451
12.120 MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait > Struct Template Reference	451

12.120.1 Member Typedef Documentation	452
12.120.1.1 Self_t	452
12.120.1.2 DelayedField_t	452
12.120.1.3 DelayedField	452
12.120.1.4 DFElt	452
12.120.2 Constructor & Destructor Documentation	452
12.120.2.1 MMHelper() [1/5]	452
12.120.2.2 MMHelper() [2/5]	453
12.120.2.3 MMHelper() [3/5]	453
12.120.2.4 MMHelper() [4/5]	453
12.120.2.5 MMHelper() [5/5]	453
12.120.3 Member Function Documentation	453
12.120.3.1 initC()	453
12.120.3.2 initA()	453
12.120.3.3 initB()	453
12.120.3.4 initOut()	453
12.120.3.5 MaxDelayedDim()	454
12.120.3.6 Aunfit()	454
12.120.3.7 Bunfit()	454
12.120.3.8 setOutBounds()	454
12.120.3.9 checkA()	454
12.120.3.10 checkB()	454
12.120.3.11 checkOut() [1/2]	454
12.120.3.12 checkOut() [2/2]	454
12.120.4 Friends And Related Symbol Documentation	455
12.120.4.1 operator<<	455
12.120.5 Field Documentation	455
12.120.5.1 recLevel	455
12.120.5.2 FieldMin	455
12.120.5.3 FieldMax	455
12.120.5.4 Amin	455
12.120.5.5 Amax	455
12.120.5.6 Bmin	455
12.120.5.7 Bmax	455
12.120.5.8 Cmin	455
12.120.5.9 Cmax	456
12.120.5.10 Outmin	456
12.120.5.11 Outmax	456
12.120.5.12 MaxStorableValue	456
12.120.5.13 delayedField	456
12.120.5.14 parseq	456
12.121 ModeTraits< Field > Struct Template Reference	456

12.121.1 Detailed Description . . . . .	456
12.121.2 Member Typedef Documentation . . . . .	456
12.121.2.1 value . . . . .	456
12.122 ModularBalanced< T > Class Template Reference . . . . .	457
12.123 ModularBalanced< T > Class Template Reference . . . . .	457
12.124 ModularTag Struct Reference . . . . .	457
12.124.1 Detailed Description . . . . .	457
12.125 Montgomery< T > Class Template Reference . . . . .	457
12.126 need_field_characteristic< Field > Struct Template Reference . . . . .	457
12.126.1 Field Documentation . . . . .	457
12.126.1.1 value . . . . .	457
12.127 NoSimd< T > Struct Template Reference . . . . .	457
12.127.1 Member Typedef Documentation . . . . .	458
12.127.1.1 vect_t . . . . .	458
12.127.1.2 scalar_t . . . . .	458
12.127.1.3 aligned_allocator . . . . .	458
12.127.1.4 aligned_vector . . . . .	458
12.127.1.5 is_same_element . . . . .	458
12.127.2 Member Function Documentation . . . . .	458
12.127.2.1 type_string() . . . . .	458
12.127.2.2 valid() . . . . .	458
12.127.2.3 compliant() . . . . .	458
12.127.3 Field Documentation . . . . .	459
12.127.3.1 vect_size . . . . .	459
12.127.3.2 alignment . . . . .	459
12.128 Parallel< C, P > Struct Template Reference . . . . .	459
12.128.1 Member Typedef Documentation . . . . .	459
12.128.1.1 Cut . . . . .	459
12.128.1.2 Param . . . . .	459
12.128.2 Constructor & Destructor Documentation . . . . .	459
12.128.2.1 Parallel() . . . . .	459
12.128.3 Member Function Documentation . . . . .	459
12.128.3.1 numthreads() . . . . .	459
12.128.3.2 set_numthreads() . . . . .	460
12.128.4 Friends And Related Symbol Documentation . . . . .	460
12.128.4.1 operator<< . . . . .	460
12.129 RNSInteger< RNS >::RandIter Class Reference . . . . .	460
12.129.1 Constructor & Destructor Documentation . . . . .	460
12.129.1.1 RandIter() . . . . .	460
12.129.2 Member Function Documentation . . . . .	460
12.129.2.1 random() [1/2] . . . . .	460
12.129.2.2 random() [2/2] . . . . .	461



12.129.2.3 operator>() [1/2]	461
12.129.2.4 operator>() [2/2]	461
12.129.2.5 ring()	461
12.130 RNSIntegerMod< RNS >::RandIter Class Reference	461
12.130.1 Constructor & Destructor Documentation	461
12.130.1.1 RandIter()	461
12.130.2 Member Function Documentation	462
12.130.2.1 random() [1/2]	462
12.130.2.2 random() [2/2]	462
12.130.2.3 operator>() [1/2]	462
12.130.2.4 operator>() [2/2]	462
12.130.2.5 ring()	462
12.131 readMyMachineType< Field, T > Struct Template Reference	462
12.131.1 Member Typedef Documentation	462
12.131.1.1 Element	462
12.131.1.2 Element_ptr	462
12.131.2 Member Function Documentation	463
12.131.2.1 operator>()	463
12.132 Recursive Struct Reference	463
12.133 Recursive Struct Reference	463
12.134 array< T >::reverse_iterator Class Reference	463
12.135 string::reverse_iterator Class Reference	463
12.136 vector< T >::reverse_iterator Class Reference	463
12.137 rint< K > Class Template Reference	463
12.138 rns_double Struct Reference	463
12.138.1 Member Typedef Documentation	464
12.138.1.1 integer	464
12.138.1.2 ModField	465
12.138.1.3 BasisElement	465
12.138.1.4 Element	465
12.138.1.5 Element_ptr	465
12.138.1.6 ConstElement_ptr	465
12.138.2 Constructor & Destructor Documentation	465
12.138.2.1 rns_double() [1/4]	465
12.138.2.2 rns_double() [2/4]	465
12.138.2.3 rns_double() [3/4]	465
12.138.2.4 rns_double() [4/4]	465
12.138.3 Member Function Documentation	465
12.138.3.1 precompute_cst()	465
12.138.3.2 init() [1/3]	466
12.138.3.3 init() [2/3]	466
12.138.3.4 init_transpose()	466

12.138.3.5	<a href="#">convert()</a> [1/2]	466
12.138.3.6	<a href="#">convert_transpose()</a>	466
12.138.3.7	<a href="#">reduce()</a>	467
12.138.3.8	<a href="#">init()</a> [3/3]	467
12.138.3.9	<a href="#">convert()</a> [2/2]	467
12.138.4	Field Documentation	467
12.138.4.1	<a href="#">_basis</a>	467
12.138.4.2	<a href="#">_basisMax</a>	467
12.138.4.3	<a href="#">_negbasis</a>	467
12.138.4.4	<a href="#">_invbasis</a>	467
12.138.4.5	<a href="#">_field_rns</a>	467
12.138.4.6	<a href="#">_M</a>	467
12.138.4.7	<a href="#">_Mi</a>	468
12.138.4.8	<a href="#">_MMi</a>	468
12.138.4.9	<a href="#">_crt_in</a>	468
12.138.4.10	<a href="#">_crt_out</a>	468
12.138.4.11	<a href="#">_size</a>	468
12.138.4.12	<a href="#">_pbits</a>	468
12.138.4.13	<a href="#">_ldm</a>	468
12.138.4.14	<a href="#">_mi_sum</a>	468
12.139	<a href="#">rns_double_elt</a> Struct Reference	468
12.139.1	Constructor & Destructor Documentation	469
12.139.1.1	<a href="#">rns_double_elt()</a> [1/3]	469
12.139.1.2	<a href="#">~rns_double_elt()</a>	469
12.139.1.3	<a href="#">rns_double_elt()</a> [2/3]	469
12.139.1.4	<a href="#">rns_double_elt()</a> [3/3]	469
12.139.2	Member Function Documentation	469
12.139.2.1	<a href="#">operator&amp;()</a> [1/2]	469
12.139.2.2	<a href="#">operator&amp;()</a> [2/2]	469
12.139.3	Field Documentation	469
12.139.3.1	<a href="#">_ptr</a>	469
12.139.3.2	<a href="#">_stride</a>	469
12.139.3.3	<a href="#">_alloc</a>	469
12.140	<a href="#">rns_double_elt_cstptr</a> Struct Reference	470
12.140.1	Constructor & Destructor Documentation	470
12.140.1.1	<a href="#">rns_double_elt_cstptr()</a> [1/5]	470
12.140.1.2	<a href="#">rns_double_elt_cstptr()</a> [2/5]	470
12.140.1.3	<a href="#">rns_double_elt_cstptr()</a> [3/5]	470
12.140.1.4	<a href="#">rns_double_elt_cstptr()</a> [4/5]	471
12.140.1.5	<a href="#">rns_double_elt_cstptr()</a> [5/5]	471
12.140.2	Member Function Documentation	471
12.140.2.1	<a href="#">operator&amp;()</a> [1/2]	471

12.140.2.2 operator*()	471
12.140.2.3 operator[]() [1/2]	471
12.140.2.4 operator[]() [2/2]	471
12.140.2.5 operator++()	471
12.140.2.6 operator--()	471
12.140.2.7 operator+()	471
12.140.2.8 operator-()	471
12.140.2.9 operator+=()	471
12.140.2.10 operator-=()	471
12.140.2.11 operator=()	472
12.140.2.12 operator<()	472
12.140.2.13 operator"!=(	472
12.140.2.14 operator&() [2/2]	472
12.140.3 Field Documentation	472
12.140.3.1 other	472
12.140.3.2 _ptr	472
12.140.3.3 _stride	472
12.140.3.4 _alloc	472
12.141 rns_double_elt_ptr Struct Reference	472
12.141.1 Constructor & Destructor Documentation	473
12.141.1.1 rns_double_elt_ptr() [1/5]	473
12.141.1.2 rns_double_elt_ptr() [2/5]	473
12.141.1.3 rns_double_elt_ptr() [3/5]	473
12.141.1.4 rns_double_elt_ptr() [4/5]	473
12.141.1.5 rns_double_elt_ptr() [5/5]	473
12.141.2 Member Function Documentation	473
12.141.2.1 operator&() [1/2]	473
12.141.2.2 operator*()	473
12.141.2.3 operator[]() [1/2]	474
12.141.2.4 operator[]() [2/2]	474
12.141.2.5 operator++()	474
12.141.2.6 operator--()	474
12.141.2.7 operator+()	474
12.141.2.8 operator-()	474
12.141.2.9 operator+=()	474
12.141.2.10 operator-=()	474
12.141.2.11 operator=()	474
12.141.2.12 operator<()	474
12.141.2.13 operator"!=(	474
12.141.2.14 operator&() [2/2]	474
12.141.3 Field Documentation	475
12.141.3.1 other	475

12.141.3.2 <code>_ptr</code>	475
12.141.3.3 <code>_stride</code>	475
12.141.3.4 <code>_alloc</code>	475
12.142 <code>rns_double_extended</code> Struct Reference	475
12.142.1 Member Typedef Documentation	476
12.142.1.1 <code>integer</code>	476
12.142.1.2 <code>ModField</code>	476
12.142.1.3 <code>BasisElement</code>	476
12.142.1.4 <code>Element</code>	476
12.142.1.5 <code>Element_ptr</code>	476
12.142.1.6 <code>ConstElement_ptr</code>	476
12.142.2 Constructor & Destructor Documentation	476
12.142.2.1 <code>rns_double_extended()</code> [1/3]	476
12.142.2.2 <code>rns_double_extended()</code> [2/3]	476
12.142.2.3 <code>rns_double_extended()</code> [3/3]	476
12.142.3 Member Function Documentation	477
12.142.3.1 <code>precompute_cst()</code>	477
12.142.3.2 <code>init()</code> [1/3]	477
12.142.3.3 <code>init()</code> [2/3]	477
12.142.3.4 <code>convert()</code> [1/2]	477
12.142.3.5 <code>init()</code> [3/3]	477
12.142.3.6 <code>convert()</code> [2/2]	477
12.142.3.7 <code>reduce()</code>	477
12.142.4 Field Documentation	478
12.142.4.1 <code>_basis</code>	478
12.142.4.2 <code>_basisMax</code>	478
12.142.4.3 <code>_negbasis</code>	478
12.142.4.4 <code>_invbasis</code>	478
12.142.4.5 <code>_field_rns</code>	478
12.142.4.6 <code>_M</code>	478
12.142.4.7 <code>_Mi</code>	478
12.142.4.8 <code>_MMi</code>	478
12.142.4.9 <code>_crt_in</code>	478
12.142.4.10 <code>_crt_out</code>	478
12.142.4.11 <code>_size</code>	478
12.142.4.12 <code>_pbits</code>	478
12.142.4.13 <code>_ldm</code>	478
12.143 <code>RNSElementTag</code> Struct Reference	479
12.143.1 Detailed Description	479
12.144 <code>RNSInteger&lt; RNS &gt;</code> Class Template Reference	479
12.144.1 Member Typedef Documentation	480
12.144.1.1 <code>BasisElement</code>	480

12.144.1.2 integer	480
12.144.1.3 Element	480
12.144.1.4 Element_ptr	480
12.144.1.5 ConstElement_ptr	480
12.144.2 Constructor & Destructor Documentation	480
12.144.2.1 RNSInteger() [1/2]	480
12.144.2.2 RNSInteger() [2/2]	480
12.144.3 Member Function Documentation	480
12.144.3.1 rns()	480
12.144.3.2 size()	480
12.144.3.3 isOne()	481
12.144.3.4 isMOne()	481
12.144.3.5 isZero()	481
12.144.3.6 characteristic()	481
12.144.3.7 cardinality()	481
12.144.3.8 init() [1/2]	481
12.144.3.9 init() [2/2]	481
12.144.3.10 reduce() [1/2]	481
12.144.3.11 reduce() [2/2]	481
12.144.3.12 convert()	482
12.144.3.13 assign()	482
12.144.3.14 write() [1/2]	482
12.144.3.15 write() [2/2]	482
12.144.4 Field Documentation	482
12.144.4.1 _rns	482
12.144.4.2 one	482
12.144.4.3 mOne	482
12.144.4.4 zero	482
12.145 RNSIntegerMod< RNS > Class Template Reference	482
12.145.1 Member Typedef Documentation	484
12.145.1.1 Element	484
12.145.1.2 Element_ptr	484
12.145.1.3 ConstElement_ptr	484
12.145.1.4 BasisElement	484
12.145.1.5 ModField	484
12.145.1.6 integer	484
12.145.2 Constructor & Destructor Documentation	484
12.145.2.1 RNSIntegerMod()	484
12.145.3 Member Function Documentation	484
12.145.3.1 rns()	484
12.145.3.2 delayed()	484
12.145.3.3 size()	485

12.145.3.4 isOne()	485
12.145.3.5 isMOne()	485
12.145.3.6 isZero()	485
12.145.3.7 characteristic() [1/2]	485
12.145.3.8 characteristic() [2/2]	485
12.145.3.9 cardinality() [1/2]	485
12.145.3.10 cardinality() [2/2]	485
12.145.3.11 minElement()	485
12.145.3.12 maxElement()	485
12.145.3.13 init() [1/3]	485
12.145.3.14 init() [2/3]	486
12.145.3.15 reduce() [1/2]	486
12.145.3.16 reduce() [2/2]	486
12.145.3.17 init() [3/3]	486
12.145.3.18 convert()	486
12.145.3.19 assign()	486
12.145.3.20 add()	486
12.145.3.21 sub()	486
12.145.3.22 neg()	487
12.145.3.23 mul()	487
12.145.3.24 axpyin()	487
12.145.3.25 inv()	487
12.145.3.26 areEqual()	487
12.145.3.27 write() [1/2]	487
12.145.3.28 write() [2/2]	487
12.145.3.29 reduce_modp() [1/2]	487
12.145.3.30 write_matrix()	488
12.145.3.31 write_matrix_long()	488
12.145.3.32 reduce_modp() [2/2]	488
12.145.3.33 reduce_modp_rnsmajor()	488
12.145.4 Field Documentation	488
12.145.4.1 _p	488
12.145.4.2 _Mi_modp_rns	488
12.145.4.3 _iM_modp_rns	488
12.145.4.4 _rns	488
12.145.4.5 _F	489
12.145.4.6 _RNSdelayed	489
12.145.4.7 one	489
12.145.4.8 mOne	489
12.145.4.9 zero	489
12.146 rnsRandIter< RNS > Class Template Reference	489
12.146.1 Constructor & Destructor Documentation	489

12.146.1.1 rnsRandIter()	489
12.146.2 Member Function Documentation	490
12.146.2.1 random() [1/2]	490
12.146.2.2 operator>() [1/2]	490
12.146.2.3 operator>() [2/2]	490
12.146.2.4 random() [2/2]	490
12.146.2.5 ring()	490
12.147 Row Struct Reference	490
12.148 ruint< K > Class Template Reference	490
12.149 ScalFunctions< Element > Struct Template Reference	490
12.149.1 Member Typedef Documentation	491
12.149.1.1 vectElt	491
12.149.2 Member Function Documentation	491
12.149.2.1 genInputs()	491
12.149.2.2 genInputsWithZero()	492
12.149.2.3 zero()	492
12.149.2.4 vand()	492
12.149.2.5 vor()	492
12.149.2.6 vxor()	492
12.149.2.7 vandnot()	492
12.149.2.8 add()	492
12.149.2.9 addin()	492
12.149.2.10 sub()	492
12.149.2.11 subin()	493
12.149.2.12 mul()	493
12.149.2.13 mulin()	493
12.149.2.14 div()	493
12.149.2.15 fmadd()	493
12.149.2.16 fmaddin()	493
12.149.2.17 fmsub()	493
12.149.2.18 fmsubin()	493
12.149.2.19 fnmadd()	494
12.149.2.20 fnmaddin()	494
12.149.2.21 lesser()	494
12.149.2.22 lesser_eq()	494
12.149.2.23 greater()	494
12.149.2.24 greater_eq()	494
12.149.2.25 eq()	494
12.149.2.26 unpacklo()	494
12.149.2.27 unpackhi()	495
12.149.2.28 unpacklohi()	495
12.149.2.29 pack_even()	495

12.149.2.30 pack_odd()	495
12.149.2.31 pack()	495
12.149.2.32 blend()	495
12.150 ScalFunctionsBase< Element, Enable > Struct Template Reference	495
12.151 Sequential Struct Reference	496
12.151.1 Constructor & Destructor Documentation	496
12.151.1.1 Sequential() [1/3]	496
12.151.1.2 Sequential() [2/3]	496
12.151.1.3 Sequential() [3/3]	496
12.151.2 Member Function Documentation	496
12.151.2.1 numthreads()	496
12.151.3 Friends And Related Symbol Documentation	496
12.151.3.1 operator<<	496
12.152 Simd128_impl< ArithType, Int, Signed, Size > Struct Template Reference	497
12.153 Simd128i_base Struct Reference	497
12.153.1 Member Typedef Documentation	497
12.153.1.1 vect_t	497
12.153.2 Member Function Documentation	497
12.153.2.1 zero()	497
12.153.2.2 sll128()	497
12.153.2.3 srl128()	497
12.153.2.4 vand()	498
12.153.2.5 vor()	498
12.153.2.6 vxor()	498
12.153.2.7 vandnot()	498
12.154 Simd256_impl< ArithType, Int, Signed, Size > Struct Template Reference	498
12.155 Simd256fp_base Struct Reference	498
12.156 Simd256i_base Struct Reference	498
12.156.1 Member Typedef Documentation	499
12.156.1.1 vect_t	499
12.156.2 Member Function Documentation	499
12.156.2.1 zero()	499
12.157 Simd512_impl< ArithType, Int, Signed, Size > Struct Template Reference	499
12.158 Simd512i_base Struct Reference	499
12.158.1 Member Typedef Documentation	499
12.158.1.1 vect_t	499
12.158.2 Member Function Documentation	500
12.158.2.1 zero()	500
12.158.2.2 vor()	500
12.158.2.3 vxor()	500
12.158.2.4 vand()	500
12.158.2.5 vandnot()	500



12.159 SimdChooser< T, bool, bool > Struct Template Reference . . . . .	500
12.160 simdToType< T > Struct Template Reference . . . . .	500
12.161 Single Struct Reference . . . . .	500
12.162 Sparse< Field, SparseMatrix_t, IdxT, PtrT > Struct Template Reference . . . . .	500
12.163 SpMat< Field, flag > Struct Template Reference . . . . .	501
12.163.1 Field Documentation . . . . .	501
12.163.1.1 _coo . . . . .	501
12.163.1.2 _csr . . . . .	501
12.163.1.3 _ell . . . . .	501
12.164 StatsMatrix Struct Reference . . . . .	501
12.164.1 Field Documentation . . . . .	502
12.164.1.1 rowdim . . . . .	502
12.164.1.2 coldim . . . . .	502
12.164.1.3 nOnes . . . . .	502
12.164.1.4 nMOnes . . . . .	502
12.164.1.5 nOthers . . . . .	502
12.164.1.6 nnz . . . . .	502
12.164.1.7 maxRow . . . . .	502
12.164.1.8 minRow . . . . .	502
12.164.1.9 averageRow . . . . .	502
12.164.1.10 deviationRow . . . . .	502
12.164.1.11 maxCol . . . . .	502
12.164.1.12 minCol . . . . .	502
12.164.1.13 averageCol . . . . .	503
12.164.1.14 deviationCol . . . . .	503
12.164.1.15 minColDifference . . . . .	503
12.164.1.16 maxColDifference . . . . .	503
12.164.1.17 averageColDifference . . . . .	503
12.164.1.18 deviationColDifference . . . . .	503
12.164.1.19 minRowDifference . . . . .	503
12.164.1.20 maxRowDifference . . . . .	503
12.164.1.21 averageRowDifference . . . . .	503
12.164.1.22 deviationRowDifference . . . . .	503
12.164.1.23 nDenseRows . . . . .	503
12.164.1.24 nDenseCols . . . . .	503
12.164.1.25 nEmptyRows . . . . .	503
12.164.1.26 nEmptyCols . . . . .	503
12.164.1.27 nEmptyColsEnd . . . . .	503
12.164.1.28 denseRows . . . . .	504
12.164.1.29 denseCols . . . . .	504
12.165 string Class Reference . . . . .	504
12.165.1 Detailed Description . . . . .	504

12.166 support_fast_mod< T > Struct Template Reference . . . . .	504
12.167 support_simd< T > Struct Template Reference . . . . .	504
12.168 support_simd_add< T > Struct Template Reference . . . . .	505
12.169 support_simd_mod< T > Struct Template Reference . . . . .	505
12.170 Test< Elt > Class Template Reference . . . . .	505
12.170.1 Member Typedef Documentation . . . . .	506
12.170.1.1 Field . . . . .	506
12.170.1.2 Elt_ptr . . . . .	506
12.170.1.3 Residu . . . . .	506
12.170.1.4 enable_if_t . . . . .	506
12.170.1.5 is_same_element . . . . .	506
12.170.1.6 enable_if_no_simd_t . . . . .	506
12.170.1.7 enable_if_simd128_t . . . . .	506
12.170.1.8 enable_if_simd256_t . . . . .	507
12.170.1.9 enable_if_simd512_t . . . . .	507
12.170.2 Constructor & Destructor Documentation . . . . .	507
12.170.2.1 Test() . . . . .	507
12.170.3 Member Function Documentation . . . . .	507
12.170.3.1 cardinality() [1/2] . . . . .	507
12.170.3.2 cardinality() [2/2] . . . . .	507
12.170.3.3 test_ftranspose() . . . . .	507
12.170.3.4 doTests() . . . . .	507
12.170.3.5 run() . . . . .	507
12.170.4 Field Documentation . . . . .	508
12.170.4.1 F . . . . .	508
12.170.4.2 _mm . . . . .	508
12.170.4.3 _nn . . . . .	508
12.171 TestOneMethod< Simd > Class Template Reference . . . . .	508
12.171.1 Member Typedef Documentation . . . . .	509
12.171.1.1 Element . . . . .	509
12.171.1.2 vect_t . . . . .	509
12.171.1.3 vectElt . . . . .	509
12.171.1.4 enable_if_t . . . . .	509
12.171.2 Constructor & Destructor Documentation . . . . .	509
12.171.2.1 TestOneMethod() . . . . .	509
12.171.3 Member Function Documentation . . . . .	509
12.171.3.1 evaluate_scalar_method() [1/3] . . . . .	509
12.171.3.2 evaluate_scalar_method() [2/3] . . . . .	510
12.171.3.3 evaluate_scalar_method() [3/3] . . . . .	510
12.171.3.4 evaluate_simd_method() [1/2] . . . . .	510
12.171.3.5 evaluate_simd_method() [2/2] . . . . .	510
12.171.3.6 getStatus() . . . . .	510

12.171.3.7	<a href="#">getTestName()</a>	510
12.171.3.8	<a href="#">writeResultLine()</a>	510
12.171.3.9	<a href="#">writeDebugData()</a>	510
12.171.4	<a href="#">Field Documentation</a>	510
12.171.4.1	<a href="#">vect_size</a>	510
12.171.4.2	<a href="#">nb_lref</a>	511
12.171.4.3	<a href="#">name</a>	511
12.171.4.4	<a href="#">inputs</a>	511
12.171.4.5	<a href="#">outputs_simd</a>	511
12.171.4.6	<a href="#">outputs_scalar</a>	511
12.172	<a href="#">tfn_minus Struct Reference</a>	511
12.172.1	<a href="#">Member Function Documentation</a>	511
12.172.1.1	<a href="#">operator&gt;()</a>	511
12.173	<a href="#">tfn_minus_eq Struct Reference</a>	511
12.173.1	<a href="#">Member Function Documentation</a>	512
12.173.1.1	<a href="#">operator&gt;()</a>	512
12.174	<a href="#">tfn_mul Struct Reference</a>	512
12.174.1	<a href="#">Member Function Documentation</a>	512
12.174.1.1	<a href="#">operator&gt;()</a>	512
12.175	<a href="#">tfn_mul_eq Struct Reference</a>	512
12.175.1	<a href="#">Member Function Documentation</a>	512
12.175.1.1	<a href="#">operator&gt;()</a>	512
12.176	<a href="#">tfn_plus Struct Reference</a>	512
12.176.1	<a href="#">Member Function Documentation</a>	513
12.176.1.1	<a href="#">operator&gt;()</a>	513
12.177	<a href="#">tfn_plus_eq Struct Reference</a>	513
12.177.1	<a href="#">Member Function Documentation</a>	513
12.177.1.1	<a href="#">operator&gt;()</a>	513
12.178	<a href="#">Threads Struct Reference</a>	513
12.179	<a href="#">ThreeD Struct Reference</a>	513
12.180	<a href="#">ThreeDAdaptive Struct Reference</a>	513
12.181	<a href="#">ThreeDInPlace Struct Reference</a>	514
12.182	<a href="#">TRSMHelper&lt; ReclterTrait, ParSeqTrait &gt; Struct Template Reference</a>	514
12.182.1	<a href="#">Detailed Description</a>	514
12.182.2	<a href="#">Constructor &amp; Destructor Documentation</a>	514
12.182.2.1	<a href="#">TRSMHelper() [1/3]</a>	514
12.182.2.2	<a href="#">TRSMHelper() [2/3]</a>	514
12.182.2.3	<a href="#">TRSMHelper() [3/3]</a>	514
12.182.3	<a href="#">Member Function Documentation</a>	515
12.182.3.1	<a href="#">pMMH() [1/2]</a>	515
12.182.3.2	<a href="#">pMMH() [2/2]</a>	515
12.182.4	<a href="#">Field Documentation</a>	515

12.182.4.1 parseq . . . . .	515
12.183 TwoD Struct Reference . . . . .	515
12.184 TwoDAdaptive Struct Reference . . . . .	515
12.185 UnparametricTag Struct Reference . . . . .	515
12.185.1 Detailed Description . . . . .	515
12.186 vector< T > Class Template Reference . . . . .	516
12.186.1 Detailed Description . . . . .	516
12.186.2 Field Documentation . . . . .	516
12.186.2.1 elements . . . . .	516
12.187 width< T > Struct Template Reference . . . . .	516
12.187.1 Field Documentation . . . . .	516
12.187.1.1 value . . . . .	516
12.188 Winograd Struct Reference . . . . .	516
12.189 WinogradPar Struct Reference . . . . .	516
<b>13 File Documentation</b>	<b>517</b>
13.1 arithprog.C File Reference . . . . .	517
13.1.1 Typedef Documentation . . . . .	517
13.1.1.1 TTimer . . . . .	517
13.1.2 Function Documentation . . . . .	517
13.1.2.1 main() . . . . .	517
13.2 autotune/charpoly.C File Reference . . . . .	517
13.2.1 Macro Definition Documentation . . . . .	518
13.2.1.1 CUBE . . . . .	518
13.2.1.2 GFOPS . . . . .	518
13.2.2 Typedef Documentation . . . . .	518
13.2.2.1 TTimer . . . . .	518
13.2.3 Function Documentation . . . . .	518
13.2.3.1 main() . . . . .	518
13.3 examples/charpoly.C File Reference . . . . .	518
13.3.1 Function Documentation . . . . .	518
13.3.1.1 main() . . . . .	518
13.4 fsyrk.C File Reference . . . . .	519
13.4.1 Macro Definition Documentation . . . . .	519
13.4.1.1 CUBE . . . . .	519
13.4.1.2 GFOPS . . . . .	519
13.4.2 Function Documentation . . . . .	519
13.4.2.1 main() . . . . .	519
13.5 fsytrf.C File Reference . . . . .	519
13.5.1 Macro Definition Documentation . . . . .	520
13.5.1.1 CUBE . . . . .	520
13.5.1.2 GFOPS . . . . .	520

13.5.2 Function Documentation	520
13.5.2.1 main()	520
13.6 ftrtri.C File Reference	520
13.6.1 Macro Definition Documentation	520
13.6.1.1 CUBE	520
13.6.1.2 GFOPS	521
13.6.2 Function Documentation	521
13.6.2.1 main()	521
13.7 autotune/pluq.C File Reference	521
13.7.1 Macro Definition Documentation	521
13.7.1.1 CUBE	521
13.7.1.2 GFOPS	521
13.7.2 Function Documentation	522
13.7.2.1 main()	522
13.8 examples/pluq.C File Reference	522
13.8.1 Function Documentation	522
13.8.1.1 main()	522
13.9 winograd.C File Reference	522
13.9.1 Macro Definition Documentation	522
13.9.1.1 DOUBLE_TO_FLOAT_CROSSOVER	522
13.9.1.2 GFOPS	523
13.9.2 Function Documentation	523
13.9.2.1 balanced() [1/2]	523
13.9.2.2 balanced() [2/2]	523
13.9.2.3 main()	523
13.10 benchmark-charpoly-mp.C File Reference	523
13.10.1 Macro Definition Documentation	523
13.10.1.1 __FFLASFFPACK_FORCE_SEQ	523
13.10.2 Function Documentation	523
13.10.2.1 main()	523
13.11 benchmark-charpoly.C File Reference	524
13.11.1 Macro Definition Documentation	524
13.11.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	524
13.11.2 Function Documentation	524
13.11.2.1 run_with_field()	524
13.11.2.2 main()	524
13.12 benchmark-checkers.C File Reference	524
13.12.1 Macro Definition Documentation	525
13.12.1.1 ENABLE_ALL_CHECKINGS	525
13.12.1.2 _NR_TESTS	525
13.12.1.3 _MAX_SIZE_MATRICES	525
13.12.1.4 CUBE	525

13.12.2 Function Documentation . . . . .	525
13.12.2.1 main() . . . . .	525
13.13 benchmark-dgemm.C File Reference . . . . .	525
13.13.1 Macro Definition Documentation . . . . .	526
13.13.1.1 CBLAS_GEMM . . . . .	526
13.13.2 Typedef Documentation . . . . .	526
13.13.2.1 TTimer . . . . .	526
13.13.2.2 Floats . . . . .	526
13.13.3 Function Documentation . . . . .	526
13.13.3.1 main() . . . . .	526
13.14 benchmark-dgetrf.C File Reference . . . . .	526
13.14.1 Macro Definition Documentation . . . . .	527
13.14.1.1 __FFLASFFPACK_HAVE_DGETRF . . . . .	527
13.14.2 Function Documentation . . . . .	527
13.14.2.1 main() . . . . .	527
13.15 benchmark-dgetri.C File Reference . . . . .	527
13.15.1 Function Documentation . . . . .	527
13.15.1.1 main() . . . . .	527
13.16 benchmark-dsytrf.C File Reference . . . . .	527
13.16.1 Macro Definition Documentation . . . . .	528
13.16.1.1 EFGFF . . . . .	528
13.16.2 Function Documentation . . . . .	528
13.16.2.1 main() . . . . .	528
13.17 benchmark-dtrsm.C File Reference . . . . .	528
13.17.1 Function Documentation . . . . .	528
13.17.1.1 main() . . . . .	528
13.18 benchmark-dtrtri.C File Reference . . . . .	528
13.18.1 Macro Definition Documentation . . . . .	529
13.18.1.1 __FFLASFFPACK_HAVE_DTRTRI . . . . .	529
13.18.2 Function Documentation . . . . .	529
13.18.2.1 main() . . . . .	529
13.19 benchmark-fadd-lvl2.C File Reference . . . . .	529
13.19.1 Macro Definition Documentation . . . . .	529
13.19.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET . . . . .	529
13.19.2 Function Documentation . . . . .	529
13.19.2.1 main() . . . . .	529
13.20 benchmark-fdot.C File Reference . . . . .	529
13.20.1 Macro Definition Documentation . . . . .	530
13.20.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET . . . . .	530
13.20.2 Function Documentation . . . . .	530
13.20.2.1 run_with_field() . . . . .	530
13.20.2.2 main() . . . . .	530

13.21 benchmark-fgemm-mp.C File Reference . . . . .	530
13.21.1 Macro Definition Documentation . . . . .	531
13.21.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET . . . . .	531
13.21.1.2 MG_DEFAULT . . . . .	531
13.21.1.3 STD_RECINT_SIZE . . . . .	531
13.21.2 Function Documentation . . . . .	531
13.21.2.1 tmain() . . . . .	531
13.21.2.2 main() . . . . .	531
13.22 benchmark-fgemm-rns.C File Reference . . . . .	531
13.22.1 Macro Definition Documentation . . . . .	532
13.22.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET . . . . .	532
13.22.2 Typedef Documentation . . . . .	532
13.22.2.1 RNS . . . . .	532
13.22.2.2 Field . . . . .	532
13.22.2.3 Element_ptr . . . . .	532
13.22.2.4 ConstElement_ptr . . . . .	532
13.22.2.5 THREADS . . . . .	532
13.22.2.6 GRAIN . . . . .	532
13.22.2.7 TWOD . . . . .	532
13.22.2.8 TWODA . . . . .	532
13.22.2.9 THREED . . . . .	532
13.22.2.10 THREEDA . . . . .	532
13.22.2.11 THREEDIP . . . . .	533
13.22.2.12 PSeq . . . . .	533
13.22.3 Function Documentation . . . . .	533
13.22.3.1 main() . . . . .	533
13.23 benchmark-fgemm.C File Reference . . . . .	533
13.23.1 Macro Definition Documentation . . . . .	533
13.23.1.1 CLASSIC_HYBRID . . . . .	533
13.23.2 Function Documentation . . . . .	533
13.23.2.1 main() . . . . .	533
13.24 benchmark-fgemv-mp.C File Reference . . . . .	533
13.24.1 Macro Definition Documentation . . . . .	534
13.24.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET . . . . .	534
13.24.1.2 MG_DEFAULT . . . . .	534
13.24.1.3 STD_RECINT_SIZE . . . . .	534
13.24.2 Function Documentation . . . . .	534
13.24.2.1 write_matrix() . . . . .	534
13.24.2.2 tmain() . . . . .	534
13.24.2.3 main() . . . . .	534
13.25 benchmark-fgemv.C File Reference . . . . .	535
13.25.1 Macro Definition Documentation . . . . .	535

13.25.1.1	<a href="#">__FFLASFFPACK_OPENBLAS_NT_ALREADY_SET</a>	535
13.25.2	Function Documentation	536
13.25.2.1	<a href="#">fill_value()</a>	536
13.25.2.2	<a href="#">genData()</a>	536
13.25.2.3	<a href="#">check_result()</a>	536
13.25.2.4	<a href="#">benchmark_with_timer()</a>	536
13.25.2.5	<a href="#">benchmark_disp()</a>	537
13.25.2.6	<a href="#">benchmark_in_Field()</a>	537
13.25.2.7	<a href="#">benchmark_with_field()</a> [1/2]	537
13.25.2.8	<a href="#">benchmark_with_field()</a> [2/2]	537
13.25.2.9	<a href="#">main()</a>	538
13.26	benchmark-fgesv.C File Reference	538
13.26.1	Macro Definition Documentation	538
13.26.1.1	<a href="#">__FFLASFFPACK_OPENBLAS_NT_ALREADY_SET</a>	538
13.26.2	Function Documentation	538
13.26.2.1	<a href="#">main()</a>	538
13.27	benchmark-fsyr2k.C File Reference	538
13.27.1	Function Documentation	539
13.27.1.1	<a href="#">main()</a>	539
13.28	benchmark-fsyrr.C File Reference	539
13.28.1	Macro Definition Documentation	539
13.28.1.1	<a href="#">__FFLASFFPACK_OPENBLAS_NT_ALREADY_SET</a>	539
13.28.2	Function Documentation	539
13.28.2.1	<a href="#">main()</a>	539
13.29	benchmark-fsytrf.C File Reference	539
13.29.1	Macro Definition Documentation	540
13.29.1.1	<a href="#">__FFPACK_FSYTRF_BC_CROUT</a>	540
13.29.1.2	<a href="#">__FFLASFFPACK_OPENBLAS_NT_ALREADY_SET</a>	540
13.29.1.3	<a href="#">CUBE</a>	540
13.29.2	Function Documentation	540
13.29.2.1	<a href="#">main()</a>	540
13.30	benchmark-ftnrm-mp.C File Reference	540
13.30.1	Macro Definition Documentation	540
13.30.1.1	<a href="#">__FFLASFFPACK_OPENBLAS_NT_ALREADY_SET</a>	540
13.30.2	Function Documentation	541
13.30.2.1	<a href="#">main()</a>	541
13.31	benchmark-ftnrm.C File Reference	541
13.31.1	Macro Definition Documentation	541
13.31.1.1	<a href="#">__FFLASFFPACK_OPENBLAS_NT_ALREADY_SET</a>	541
13.31.2	Function Documentation	541
13.31.2.1	<a href="#">main()</a>	541
13.32	benchmark-ftnrv.C File Reference	541



13.32.1 Macro Definition Documentation	542
13.32.1.1 __FflasFfpack_OPENBLAS_NT_ALREADY_SET	542
13.32.2 Function Documentation	542
13.32.2.1 main()	542
13.33 benchmark-ftsrti.C File Reference	542
13.33.1 Macro Definition Documentation	542
13.33.1.1 __FflasFfpack_OPENBLAS_NT_ALREADY_SET	542
13.33.1.2 CUBE	542
13.33.2 Function Documentation	542
13.33.2.1 main()	542
13.34 benchmark-inverse.C File Reference	542
13.34.1 Macro Definition Documentation	543
13.34.1.1 CUBE	543
13.34.2 Function Documentation	543
13.34.2.1 main()	543
13.35 benchmark-lqmp.C File Reference	543
13.35.1 Function Documentation	543
13.35.1.1 main()	543
13.36 benchmark-lqmp.C File Reference	544
13.36.1 Macro Definition Documentation	544
13.36.1.1 CUBE	544
13.36.2 Function Documentation	544
13.36.2.1 main()	544
13.37 benchmark-pluq.C File Reference	544
13.37.1 Macro Definition Documentation	545
13.37.1.1 __FflasFfpack_OPENBLAS_NT_ALREADY_SET	545
13.37.1.2 CUBE	545
13.37.2 Typedef Documentation	545
13.37.2.1 Field	545
13.37.3 Function Documentation	545
13.37.3.1 verification_PLUQ()	545
13.37.3.2 Rec_Initialize()	545
13.37.3.3 main()	545
13.38 benchmark-quasisep.C File Reference	545
13.38.1 Macro Definition Documentation	546
13.38.1.1 __FflasFfpack_OPENBLAS_NT_ALREADY_SET	546
13.38.2 Function Documentation	546
13.38.2.1 run_with_field()	546
13.38.2.2 main()	546
13.39 benchmark-storage-transpose.C File Reference	546
13.39.1 Function Documentation	547
13.39.1.1 main()	547

13.40 benchmark-wino.C File Reference	547
13.40.1 Macro Definition Documentation	547
13.40.1.1 CUBE	547
13.40.2 Function Documentation	547
13.40.2.1 launch_wino()	547
13.40.2.2 main()	548
13.41 mainpage.doxy File Reference	548
13.42 det.C File Reference	548
13.42.1 Function Documentation	548
13.42.1.1 main()	548
13.43 matmul.C File Reference	548
13.43.1 Function Documentation	548
13.43.1.1 main()	548
13.44 rank.C File Reference	549
13.44.1 Function Documentation	549
13.44.1.1 main()	549
13.45 solve.C File Reference	549
13.45.1 Function Documentation	549
13.45.1.1 main()	549
13.46 checker_charpoly.inl File Reference	549
13.46.1 Macro Definition Documentation	550
13.46.1.1 __FFLASFFPACK_checker_charpoly_INL	550
13.47 checker_det.inl File Reference	550
13.47.1 Macro Definition Documentation	550
13.47.1.1 __FFLASFFPACK_checker_det_INL	550
13.48 checker_empty.h File Reference	550
13.49 checker_fgemm.inl File Reference	550
13.49.1 Macro Definition Documentation	551
13.49.1.1 __FFLASFFPACK_checker_fgemm_INL	551
13.50 checker_ftsm.inl File Reference	551
13.50.1 Macro Definition Documentation	551
13.50.1.1 __FFLASFFPACK_checker_ftsm_INL	551
13.51 checker_invert.inl File Reference	551
13.51.1 Macro Definition Documentation	551
13.51.1.1 __FFLASFFPACK_checker_invert_INL	551
13.52 checker_pluq.inl File Reference	551
13.52.1 Macro Definition Documentation	552
13.52.1.1 __FFLASFFPACK_checker_pluq_INL	552
13.53 checkers.doxy File Reference	552
13.54 checkers_fflas.h File Reference	552
13.55 checkers_fflas.inl File Reference	552
13.55.1 Macro Definition Documentation	553

13.55.1.1 FFLASFFPACK_checkers_fflas_inl_H . . . . .	553
13.56 checkers_ffpack.h File Reference . . . . .	553
13.57 checkers_ffpack.inl File Reference . . . . .	553
13.57.1 Macro Definition Documentation . . . . .	554
13.57.1.1 FFLASFFPACK_checkers_ffpack_inl_H . . . . .	554
13.58 config-blas.h File Reference . . . . .	554
13.58.1 Macro Definition Documentation . . . . .	555
13.58.1.1 CBLAS_INT . . . . .	555
13.58.1.2 CBLAS_ENUM_DEFINED_H . . . . .	555
13.58.1.3 CBLAS_EXTERNALS . . . . .	555
13.58.1.4 blas_enum . . . . .	555
13.58.2 Enumeration Type Documentation . . . . .	555
13.58.2.1 CBLAS_ORDER . . . . .	555
13.58.2.2 CBLAS_TRANSPOSE . . . . .	555
13.58.2.3 CBLAS_UPLO . . . . .	556
13.58.2.4 CBLAS_DIAG . . . . .	556
13.58.2.5 CBLAS_SIDE . . . . .	556
13.58.3 Function Documentation . . . . .	556
13.58.3.1 daxpy_() . . . . .	556
13.58.3.2 saxpy_() . . . . .	556
13.58.3.3 ddot_() . . . . .	557
13.58.3.4 sdot_() . . . . .	557
13.58.3.5 dasum_() . . . . .	557
13.58.3.6 idamax_() . . . . .	557
13.58.3.7 dnrm2_() . . . . .	557
13.58.3.8 dgemv_() . . . . .	557
13.58.3.9 sgemv_() . . . . .	557
13.58.3.10 dger_() . . . . .	558
13.58.3.11 sger_() . . . . .	558
13.58.3.12 dcopy_() . . . . .	558
13.58.3.13 scopy_() . . . . .	558
13.58.3.14 dscal_() . . . . .	558
13.58.3.15 sscal_() . . . . .	559
13.58.3.16 dtrsm_() . . . . .	559
13.58.3.17 strsm_() . . . . .	559
13.58.3.18 dtrmm_() . . . . .	559
13.58.3.19 strmm_() . . . . .	559
13.58.3.20 sgemm_() . . . . .	560
13.58.3.21 dgemm_() . . . . .	560
13.58.3.22 cblas_dsyrk() . . . . .	560
13.59 config.h File Reference . . . . .	561
13.59.1 Macro Definition Documentation . . . . .	561

13.59.1.1 HAVE_BLAS . . . . .	561
13.59.1.2 HAVE_CBLAS . . . . .	561
13.59.1.3 HAVE_CXX11 . . . . .	562
13.59.1.4 HAVE_DLFCN_H . . . . .	562
13.59.1.5 HAVE_FLOAT_H . . . . .	562
13.59.1.6 HAVE_INT128 . . . . .	562
13.59.1.7 HAVE_INTPTR_T . . . . .	562
13.59.1.8 HAVE_LAPACK . . . . .	562
13.59.1.9 HAVE_LIMITS_H . . . . .	562
13.59.1.10 HAVE_LITTLE_ENDIAN . . . . .	562
13.59.1.11 HAVE_PTHREAD_H . . . . .	562
13.59.1.12 HAVE_STDDEF_H . . . . .	562
13.59.1.13 HAVE_STDINT_H . . . . .	562
13.59.1.14 HAVE_STDIO_H . . . . .	562
13.59.1.15 HAVE_STDLIB_H . . . . .	562
13.59.1.16 HAVE_STRINGS_H . . . . .	562
13.59.1.17 HAVE_STRING_H . . . . .	562
13.59.1.18 HAVE_SYS_STAT_H . . . . .	563
13.59.1.19 HAVE_SYS_TIME_H . . . . .	563
13.59.1.20 HAVE_SYS_TYPES_H . . . . .	563
13.59.1.21 HAVE_UNISTD_H . . . . .	563
13.59.1.22 LT_OBJDIR . . . . .	563
13.59.1.23 OPENBLAS_NUM_THREADS . . . . .	563
13.59.1.24 PACKAGE . . . . .	563
13.59.1.25 PACKAGE_BUGREPORT . . . . .	563
13.59.1.26 PACKAGE_NAME . . . . .	563
13.59.1.27 PACKAGE_STRING . . . . .	563
13.59.1.28 PACKAGE_TARNAME . . . . .	563
13.59.1.29 PACKAGE_URL . . . . .	563
13.59.1.30 PACKAGE_VERSION . . . . .	563
13.59.1.31 SIZEOF_CHAR . . . . .	563
13.59.1.32 SIZEOF_INT . . . . .	563
13.59.1.33 SIZEOF_LONG . . . . .	564
13.59.1.34 SIZEOF_LONG_LONG . . . . .	564
13.59.1.35 SIZEOF_SHORT . . . . .	564
13.59.1.36 SIZEOF__INT64_T . . . . .	564
13.59.1.37 STDC_HEADERS . . . . .	564
13.59.1.38 USE_OPENMP . . . . .	564
13.59.1.39 VERSION . . . . .	564
13.60 fflas-ffpack/config.h File Reference . . . . .	564
13.60.1 Macro Definition Documentation . . . . .	565
13.60.1.1 __FFLASFFPACK_HAVE_BLAS . . . . .	565

13.60.1.2	<a href="#">__FFLASFFPACK_HAVE_CBLAS</a>	565
13.60.1.3	<a href="#">__FFLASFFPACK_HAVE_CXX11</a>	565
13.60.1.4	<a href="#">__FFLASFFPACK_HAVE_DLFCN_H</a>	565
13.60.1.5	<a href="#">__FFLASFFPACK_HAVE_FLOAT_H</a>	565
13.60.1.6	<a href="#">__FFLASFFPACK_HAVE_INT128</a>	565
13.60.1.7	<a href="#">__FFLASFFPACK_HAVE_INTPTR_T</a>	565
13.60.1.8	<a href="#">__FFLASFFPACK_HAVE_LAPACK</a>	565
13.60.1.9	<a href="#">__FFLASFFPACK_HAVE_LIMITS_H</a>	565
13.60.1.10	<a href="#">__FFLASFFPACK_HAVE_LITTLE_ENDIAN</a>	565
13.60.1.11	<a href="#">__FFLASFFPACK_HAVE_PTHREAD_H</a>	565
13.60.1.12	<a href="#">__FFLASFFPACK_HAVE_STDDEF_H</a>	566
13.60.1.13	<a href="#">__FFLASFFPACK_HAVE_STDINT_H</a>	566
13.60.1.14	<a href="#">__FFLASFFPACK_HAVE_STDIO_H</a>	566
13.60.1.15	<a href="#">__FFLASFFPACK_HAVE_STDLIB_H</a>	566
13.60.1.16	<a href="#">__FFLASFFPACK_HAVE_STRINGS_H</a>	566
13.60.1.17	<a href="#">__FFLASFFPACK_HAVE_STRING_H</a>	566
13.60.1.18	<a href="#">__FFLASFFPACK_HAVE_SYS_STAT_H</a>	566
13.60.1.19	<a href="#">__FFLASFFPACK_HAVE_SYS_TIME_H</a>	566
13.60.1.20	<a href="#">__FFLASFFPACK_HAVE_SYS_TYPES_H</a>	566
13.60.1.21	<a href="#">__FFLASFFPACK_HAVE_UNISTD_H</a>	566
13.60.1.22	<a href="#">__FFLASFFPACK_LT_OBJDIR</a>	566
13.60.1.23	<a href="#">__FFLASFFPACK_OPENBLAS_NUM_THREADS</a>	566
13.60.1.24	<a href="#">__FFLASFFPACK_PACKAGE</a>	566
13.60.1.25	<a href="#">__FFLASFFPACK_PACKAGE_BUGREPORT</a>	566
13.60.1.26	<a href="#">__FFLASFFPACK_PACKAGE_NAME</a>	566
13.60.1.27	<a href="#">__FFLASFFPACK_PACKAGE_STRING</a>	567
13.60.1.28	<a href="#">__FFLASFFPACK_PACKAGE_TARNAME</a>	567
13.60.1.29	<a href="#">__FFLASFFPACK_PACKAGE_URL</a>	567
13.60.1.30	<a href="#">__FFLASFFPACK_PACKAGE_VERSION</a>	567
13.60.1.31	<a href="#">__FFLASFFPACK_SIZEOF_CHAR</a>	567
13.60.1.32	<a href="#">__FFLASFFPACK_SIZEOF_INT</a>	567
13.60.1.33	<a href="#">__FFLASFFPACK_SIZEOF_LONG</a>	567
13.60.1.34	<a href="#">__FFLASFFPACK_SIZEOF_LONG_LONG</a>	567
13.60.1.35	<a href="#">__FFLASFFPACK_SIZEOF_SHORT</a>	567
13.60.1.36	<a href="#">__FFLASFFPACK_SIZEOF__INT64_T</a>	567
13.60.1.37	<a href="#">__FFLASFFPACK_STDC_HEADERS</a>	567
13.60.1.38	<a href="#">__FFLASFFPACK_USE_OPENMP</a>	567
13.60.1.39	<a href="#">__FFLASFFPACK_VERSION</a>	567
13.61	<a href="#">fflas-ffpack-config.h File Reference</a>	567
13.61.1	<a href="#">Detailed Description</a>	568
13.61.2	<a href="#">Macro Definition Documentation</a>	568
13.61.2.1	<a href="#">GCC_VERSION</a>	568

13.62 fflas-ffpack-default-thresholds.h File Reference	568
13.62.1 Macro Definition Documentation	568
13.62.1.1 __FFLASFFPACK_WINOTHRESHOLD	568
13.62.1.2 __FFLASFFPACK_WINOTHRESHOLD_FLT	568
13.62.1.3 __FFLASFFPACK_WINOTHRESHOLD_BAL	568
13.62.1.4 __FFLASFFPACK_WINOTHRESHOLD_BAL_FLT	568
13.62.1.5 __FFLASFFPACK_PLUQ_THRESHOLD	568
13.62.1.6 __FFLASFFPACK_CHARPOLY_LUKrylov_ArithProg_THRESHOLD	568
13.62.1.7 __FFLASFFPACK_CHARPOLY_Danilevskii_LUKrylov_THRESHOLD	569
13.62.1.8 __FFLASFFPACK_ARITHPROG_THRESHOLD	569
13.62.1.9 __FFLASFFPACK_FTRTRI_THRESHOLD	569
13.62.1.10 __FFLASFFPACK_FSYTRF_THRESHOLD	569
13.62.1.11 __FFLASFFPACK_FSYRK_THRESHOLD	569
13.63 fflas-ffpack-thresholds.h File Reference	569
13.64 fflas-ffpack.doxy File Reference	569
13.65 fflas-ffpack.h File Reference	569
13.65.1 Detailed Description	569
13.66 fflas.doxy File Reference	569
13.67 fflas.h File Reference	569
13.67.1 Detailed Description	570
13.67.2 Macro Definition Documentation	570
13.67.2.1 WINOTHRESHOLD	570
13.67.2.2 DOUBLE_TO_FLOAT_CROSSOVER	570
13.68 fflas_bounds.inl File Reference	571
13.68.1 Macro Definition Documentation	571
13.68.1.1 __FFLASFFPACK_fflas_bounds_INL	571
13.68.1.2 FFLAS_INT_TYPE	571
13.69 fflas_enum.h File Reference	571
13.70 fflas_fadd.h File Reference	572
13.71 fflas_fadd.inl File Reference	573
13.71.1 Macro Definition Documentation	574
13.71.1.1 __FFLASFFPACK_fadd_INL	574
13.72 fflas_fassign.h File Reference	574
13.73 fflas_fassign.inl File Reference	575
13.73.1 Macro Definition Documentation	575
13.73.1.1 __FFLASFFPACK_fassign_INL	575
13.74 fflas_faxpy.inl File Reference	575
13.74.1 Macro Definition Documentation	576
13.74.1.1 __FFLASFFPACK_faxpy_INL	576
13.75 fflas_fdot.inl File Reference	576
13.75.1 Macro Definition Documentation	577
13.75.1.1 __FFLASFFPACK_fdot_INL	577

13.76 fflas_fgemm.inl File Reference . . . . .	577
13.76.1 Macro Definition Documentation . . . . .	579
13.76.1.1 __FFLASFFPACK_fgemm_INL . . . . .	579
13.77 fgemm_classical.inl File Reference . . . . .	579
13.78 fgemm_classical_mp.inl File Reference . . . . .	579
13.78.1 Detailed Description . . . . .	581
13.78.2 Macro Definition Documentation . . . . .	581
13.78.2.1 __FFPACK_fgemm_classical_INL . . . . .	581
13.79 fgemm_winograd.inl File Reference . . . . .	581
13.79.1 Macro Definition Documentation . . . . .	583
13.79.1.1 __FFLASFFPACK_fflas_fflas_fgemm_winograd_INL . . . . .	583
13.79.1.2 NEWWINO . . . . .	583
13.80 matmul.doxy File Reference . . . . .	583
13.81 schedule_bini.inl File Reference . . . . .	583
13.81.1 Detailed Description . . . . .	583
13.81.2 Macro Definition Documentation . . . . .	583
13.81.2.1 __FFLASFFPACK_fgemm_bini_INL . . . . .	583
13.82 schedule_winograd.inl File Reference . . . . .	583
13.82.1 Macro Definition Documentation . . . . .	584
13.82.1.1 __FFLASFFPACK_fgemm_winograd_INL . . . . .	584
13.83 schedule_winograd_acc.inl File Reference . . . . .	584
13.83.1 Macro Definition Documentation . . . . .	585
13.83.1.1 __FFLASFFPACK_fgemm_winograd_acc_INL . . . . .	585
13.84 schedule_winograd_acc_ip.inl File Reference . . . . .	585
13.84.1 Macro Definition Documentation . . . . .	585
13.84.1.1 __FFLASFFPACK_fgemm_winograd_acc_ip_INL . . . . .	585
13.85 schedule_winograd_ip.inl File Reference . . . . .	585
13.85.1 Macro Definition Documentation . . . . .	586
13.85.1.1 __FFLASFFPACK_fgemm_winograd_ip_INL . . . . .	586
13.86 fflas_fgemv.inl File Reference . . . . .	586
13.86.1 Macro Definition Documentation . . . . .	587
13.86.1.1 __FFLASFFPACK_fgemv_INL . . . . .	587
13.87 fflas_fgemv_mp.inl File Reference . . . . .	588
13.87.1 Macro Definition Documentation . . . . .	588
13.87.1.1 __FFLASFFPACK_fgemv_mp_INL . . . . .	588
13.88 fflas_fger.inl File Reference . . . . .	588
13.88.1 Macro Definition Documentation . . . . .	589
13.88.1.1 __FFLASFFPACK_fger_INL . . . . .	589
13.89 fflas_fger_mp.inl File Reference . . . . .	589
13.89.1 Macro Definition Documentation . . . . .	590
13.89.1.1 __FFPACK_fger_mp_INL . . . . .	590
13.90 fflas_freduce.h File Reference . . . . .	590

13.91 fflas_freduce.inl File Reference . . . . .	591
13.91.1 Macro Definition Documentation . . . . .	593
13.91.1.1 __FFLASFFPACK_fflas_freduce_INL . . . . .	593
13.91.1.2 FFLASFFPACK_COPY_REDUCE . . . . .	593
13.92 fflas_freduce_mp.inl File Reference . . . . .	593
13.92.1 Macro Definition Documentation . . . . .	593
13.92.1.1 __FFLASFFPACK_fflas_freduce_mp_INL . . . . .	593
13.93 fflas_freivalds.inl File Reference . . . . .	593
13.93.1 Macro Definition Documentation . . . . .	593
13.93.1.1 __FFLASFFPACK_freivalds_INL . . . . .	593
13.94 fflas_fscal.h File Reference . . . . .	594
13.95 fflas_fscal.inl File Reference . . . . .	594
13.95.1 Macro Definition Documentation . . . . .	595
13.95.1.1 __FFLASFFPACK_fscal_INL . . . . .	595
13.96 fflas_fscal_mp.inl File Reference . . . . .	595
13.96.1 Macro Definition Documentation . . . . .	596
13.96.1.1 __FFLASFFPACK_fscal_mp_INL . . . . .	596
13.97 fflas_fsyr2k.inl File Reference . . . . .	596
13.97.1 Macro Definition Documentation . . . . .	596
13.97.1.1 __FFLASFFPACK_fflas_fsyr2k_INL . . . . .	596
13.98 fflas_fsyrk.inl File Reference . . . . .	597
13.98.1 Macro Definition Documentation . . . . .	598
13.98.1.1 __FFLASFFPACK_fflas_fsyrk_INL . . . . .	598
13.99 fflas_fsyrk_strassen.inl File Reference . . . . .	598
13.99.1 Macro Definition Documentation . . . . .	599
13.99.1.1 __FFLASFFPACK_fflas_fsyrk_strassen_INL . . . . .	599
13.100 fflas_ftrmm.inl File Reference . . . . .	599
13.100.1 Macro Definition Documentation . . . . .	600
13.100.1.1 __FFLASFFPACK_ftrmm_INL . . . . .	600
13.101 fflas_ftrsm.inl File Reference . . . . .	600
13.101.1 Macro Definition Documentation . . . . .	600
13.101.1.1 __FFLASFFPACK_ftrsm_INL . . . . .	600
13.102 fflas_ftrsm_mp.inl File Reference . . . . .	601
13.102.1 Detailed Description . . . . .	601
13.102.2 Macro Definition Documentation . . . . .	601
13.102.2.1 __FFPACK_ftrsm_mp_INL . . . . .	601
13.103 fflas_ftrsv.inl File Reference . . . . .	601
13.103.1 Macro Definition Documentation . . . . .	602
13.103.1.1 __FFLASFFPACK_ftrsv_INL . . . . .	602
13.104 fflas_helpers.inl File Reference . . . . .	602
13.104.1 Macro Definition Documentation . . . . .	603
13.104.1.1 __FFLASFFPACK_fflas_mmmhelper_INL . . . . .	603



13.105 igemm.doxy File Reference . . . . .	603
13.106 igemm.h File Reference . . . . .	603
13.107 igemm.inl File Reference . . . . .	603
13.107.1 Macro Definition Documentation . . . . .	604
13.107.1.1 __FFLASFFPACK_fflas_igemm_igemm_INL . . . . .	604
13.108 igemm_kernels.h File Reference . . . . .	604
13.109 igemm_kernels.inl File Reference . . . . .	605
13.109.1 Macro Definition Documentation . . . . .	605
13.109.1.1 __FFLASFFPACK_fflas_igemm_igemm_kernels_INL . . . . .	605
13.110 igemm_tools.h File Reference . . . . .	605
13.111 igemm_tools.inl File Reference . . . . .	606
13.111.1 Macro Definition Documentation . . . . .	606
13.111.1.1 __FFLASFFPACK_fflas_igemm_igemm_tools_INL . . . . .	606
13.112 fflas_level1.inl File Reference . . . . .	606
13.112.1 Macro Definition Documentation . . . . .	608
13.112.1.1 __FFLASFFPACK_fflas_fflas_level1_INL . . . . .	608
13.113 fflas_level2.inl File Reference . . . . .	609
13.113.1 Macro Definition Documentation . . . . .	611
13.113.1.1 __FFLASFFPACK_fflas_fflas_level2_INL . . . . .	611
13.114 fflas_level3.inl File Reference . . . . .	611
13.114.1 Macro Definition Documentation . . . . .	614
13.114.1.1 __FFLASFFPACK_fflas_fflas_level3_INL . . . . .	614
13.114.1.2 __FFLAS__TRSM_READONLY . . . . .	614
13.115 fflas_pfgemm.inl File Reference . . . . .	614
13.115.1 Macro Definition Documentation . . . . .	614
13.115.1.1 __FFLASFFPACK_fflas_pfgemm_INL . . . . .	614
13.115.1.2 __FFLASFFPACK_SEQPARTHRESHOLD . . . . .	614
13.115.1.3 __FFLASFFPACK_DIMKPENALTY . . . . .	614
13.116 fflas_pftsm.inl File Reference . . . . .	615
13.116.1 Macro Definition Documentation . . . . .	615
13.116.1.1 __FFLASFFPACK_fflas_pftsm_INL . . . . .	615
13.116.1.2 PTRSM_HYBRID_THRESHOLD . . . . .	615
13.117 fflas_simd.h File Reference . . . . .	615
13.117.1 Macro Definition Documentation . . . . .	616
13.117.1.1 SIMD_INT . . . . .	616
13.117.1.2 INLINE . . . . .	616
13.117.1.3 CONST . . . . .	616
13.117.1.4 PURE . . . . .	616
13.117.1.5 NORML_MOD . . . . .	616
13.117.1.6 FLOAT_MOD . . . . .	617
13.117.2 Typedef Documentation . . . . .	617
13.117.2.1 Simd . . . . .	617

13.118 simd.doxy File Reference . . . . .	617
13.119 simd128.inl File Reference . . . . .	617
13.119.1 Macro Definition Documentation . . . . .	617
13.119.1.1 __FFLASFFPACK_fflas_ffpack_utils_simd128_INL . . . . .	617
13.119.2 Typedef Documentation . . . . .	618
13.119.2.1 Simd128 . . . . .	618
13.120 simd128_double.inl File Reference . . . . .	618
13.120.1 Macro Definition Documentation . . . . .	618
13.120.1.1 __FFLASFFPACK_fflas_ffpack_utils_simd128_double_INL . . . . .	618
13.121 simd128_float.inl File Reference . . . . .	618
13.121.1 Macro Definition Documentation . . . . .	618
13.121.1.1 __FFLASFFPACK_fflas_ffpack_utils_simd128_float_INL . . . . .	618
13.122 simd128_int16.inl File Reference . . . . .	618
13.122.1 Macro Definition Documentation . . . . .	619
13.122.1.1 __FFLASFFPACK_fflas_ffpack_utils_simd128_int16_INL . . . . .	619
13.123 simd128_int32.inl File Reference . . . . .	619
13.123.1 Macro Definition Documentation . . . . .	619
13.123.1.1 __FFLASFFPACK_fflas_ffpack_utils_simd128_int32_INL . . . . .	619
13.124 simd128_int64.inl File Reference . . . . .	619
13.124.1 Macro Definition Documentation . . . . .	620
13.124.1.1 __FFLASFFPACK_fflas_ffpack_utils_simd128_int64_INL . . . . .	620
13.124.1.2 vect_t . . . . .	620
13.125 simd256.inl File Reference . . . . .	620
13.125.1 Macro Definition Documentation . . . . .	620
13.125.1.1 __FFLASFFPACK_fflas_ffpack_utils_simd256_INL . . . . .	620
13.125.2 Typedef Documentation . . . . .	620
13.125.2.1 Simd256 . . . . .	620
13.126 simd256_double.inl File Reference . . . . .	621
13.126.1 Macro Definition Documentation . . . . .	621
13.126.1.1 __FFLASFFPACK_fflas_ffpack_utils_simd256_double_INL . . . . .	621
13.127 simd256_float.inl File Reference . . . . .	621
13.127.1 Macro Definition Documentation . . . . .	621
13.127.1.1 __FFLASFFPACK_fflas_ffpack_utils_simd256_float_INL . . . . .	621
13.128 simd256_int16.inl File Reference . . . . .	621
13.128.1 Macro Definition Documentation . . . . .	622
13.128.1.1 __FFLASFFPACK_fflas_ffpack_utils_simd256_int16_INL . . . . .	622
13.129 simd256_int32.inl File Reference . . . . .	622
13.129.1 Macro Definition Documentation . . . . .	622
13.129.1.1 __FFLASFFPACK_fflas_ffpack_utils_simd256_int32_INL . . . . .	622
13.130 simd256_int64.inl File Reference . . . . .	622
13.130.1 Macro Definition Documentation . . . . .	623
13.130.1.1 __FFLASFFPACK_fflas_ffpack_utils_simd256_int64_INL . . . . .	623

13.130.1.2 vect_t	623
13.131 simd512.inl File Reference	623
13.131.1 Macro Definition Documentation	623
13.131.1.1 __FFLASFFPACK_simd512_INL	623
13.131.2 Typedef Documentation	623
13.131.2.1 Simd512	623
13.132 simd512_double.inl File Reference	623
13.132.1 Macro Definition Documentation	624
13.132.1.1 __FFLASFFPACK_simd512_double_INL	624
13.133 simd512_float.inl File Reference	624
13.133.1 Macro Definition Documentation	624
13.133.1.1 __FFLASFFPACK_simd512_float_INL	624
13.134 simd512_int32.inl File Reference	624
13.134.1 Macro Definition Documentation	624
13.134.1.1 __FFLASFFPACK_simd512_int32_INL	624
13.135 simd512_int64.inl File Reference	625
13.135.1 Macro Definition Documentation	625
13.135.1.1 _simd512_int64_INL	625
13.135.1.2 vect_t	625
13.136 simd_modular.inl File Reference	625
13.137 fflas_sparse.h File Reference	625
13.137.1 Macro Definition Documentation	629
13.137.1.1 index_t	629
13.137.1.2 ROUND_DOWN	629
13.137.1.3 __FFLASFFPACK_CACHE_LINE_SIZE	629
13.137.1.4 assume_aligned	629
13.137.1.5 DENSE_THRESHOLD	629
13.138 fflas_sparse.inl File Reference	630
13.138.1 Macro Definition Documentation	631
13.138.1.1 __FFLASFFPACK_fflas_fflas_sparse_INL	631
13.139 coo.h File Reference	632
13.140 coo_spmv.inl File Reference	632
13.140.1 Macro Definition Documentation	633
13.140.1.1 __FFLASFFPACK_fflas_sparse_coo_spmv_INL	633
13.141 coo_spmv.inl File Reference	633
13.141.1 Macro Definition Documentation	634
13.141.1.1 __FFLASFFPACK_fflas_sparse_coo_spmv_INL	634
13.142 coo_utils.inl File Reference	634
13.142.1 Macro Definition Documentation	634
13.142.1.1 __FFLASFFPACK_fflas_sparse_coo_utils_INL	634
13.143 csr.h File Reference	634
13.144 csr_pspmm.inl File Reference	635

13.144.1 Macro Definition Documentation	636
13.144.1.1 __FFLASFFPACK_fflas_sparse_CSR_pspmm_INL	636
13.145 csr_pspmv.inl File Reference	636
13.145.1 Macro Definition Documentation	636
13.145.1.1 __FFLASFFPACK_fflas_sparse_CSR_pspmv_INL	636
13.146 csr_spm্ম.inl File Reference	636
13.146.1 Macro Definition Documentation	637
13.146.1.1 __FFLASFFPACK_fflas_sparse_CSR_spm্ম_INL	637
13.147 csr_spmv.inl File Reference	638
13.147.1 Macro Definition Documentation	638
13.147.1.1 __FFLASFFPACK_fflas_sparse_CSR_spmv_INL	638
13.148 csr_utils.inl File Reference	638
13.149 csr_hyb.h File Reference	639
13.150 csr_hyb_pspmm.inl File Reference	639
13.150.1 Macro Definition Documentation	640
13.150.1.1 __FFLASFFPACK_fflas_sparse_CSR_HYB_pspmm_INL	640
13.151 csr_hyb_pspmv.inl File Reference	640
13.151.1 Macro Definition Documentation	640
13.151.1.1 __FFLASFFPACK_fflas_sparse_CSR_HYB_pspmv_INL	640
13.152 csr_hyb_spm্ম.inl File Reference	641
13.152.1 Macro Definition Documentation	641
13.152.1.1 __FFLASFFPACK_fflas_sparse_CSR_HYB_spm্ম_INL	641
13.153 csr_hyb_spmv.inl File Reference	641
13.153.1 Macro Definition Documentation	641
13.153.1.1 __FFLASFFPACK_fflas_sparse_CSR_HYB_spmv_INL	641
13.154 csr_hyb_utils.inl File Reference	642
13.154.1 Macro Definition Documentation	642
13.154.1.1 __FFLASFFPACK_fflas_sparse_CSR_HYB_utils_INL	642
13.155 ell.h File Reference	642
13.156 ell_pspmm.inl File Reference	643
13.156.1 Macro Definition Documentation	643
13.156.1.1 __FFLASFFPACK_fflas_sparse_ELL_pspmm_INL	643
13.157 ell_pspmv.inl File Reference	643
13.157.1 Macro Definition Documentation	644
13.157.1.1 __FFLASFFPACK_fflas_sparse_ELL_pspmv_INL	644
13.158 ell_spm্ম.inl File Reference	644
13.158.1 Macro Definition Documentation	645
13.158.1.1 __FFLASFFPACK_fflas_sparse_ELL_spm্ম_INL	645
13.159 ell_spmv.inl File Reference	645
13.159.1 Macro Definition Documentation	646
13.159.1.1 __FFLASFFPACK_fflas_sparse_ELL_spmv_INL	646
13.160 ell_utils.inl File Reference	646

13.160.1 Macro Definition Documentation	646
13.160.1.1 __FFLASFFPACK_fflas_sparse_ELL_utils_INL	646
13.161 ell_simd.h File Reference	646
13.162 ell_simd_pspmv.inl File Reference	647
13.162.1 Macro Definition Documentation	648
13.162.1.1 __FFLASFFPACK_fflas_sparse_ELL_simd_pspmv_INL	648
13.163 ell_simd_spmv.inl File Reference	648
13.163.1 Macro Definition Documentation	648
13.163.1.1 __FFLASFFPACK_fflas_sparse_ELL_simd_spmv_INL	648
13.164 ell_simd_utils.inl File Reference	649
13.164.1 Macro Definition Documentation	649
13.164.1.1 __FFLASFFPACK_fflas_sparse_ELL_simd_utils_INL	649
13.165 hyb_zo.h File Reference	649
13.166 hyb_zo_pspmm.inl File Reference	649
13.166.1 Macro Definition Documentation	650
13.166.1.1 __FFLASFFPACK_fflas_sparse_HYB_ZO_pspmm_INL	650
13.167 hyb_zo_pspmv.inl File Reference	650
13.167.1 Macro Definition Documentation	650
13.167.1.1 __FFLASFFPACK_fflas_sparse_HYB_ZO_pspmv_INL	650
13.168 hyb_zo_spmmm.inl File Reference	650
13.168.1 Macro Definition Documentation	651
13.168.1.1 __FFLASFFPACK_fflas_sparse_HYB_ZO_spmmm_INL	651
13.169 hyb_zo_spmv.inl File Reference	651
13.169.1 Macro Definition Documentation	651
13.169.1.1 __FFLASFFPACK_fflas_sparse_HYB_ZO_spmv_INL	651
13.170 hyb_zo_utils.inl File Reference	651
13.170.1 Macro Definition Documentation	652
13.170.1.1 __FFLASFFPACK_fflas_sparse_HYB_ZO_utils_INL	652
13.171 read_sparse.h File Reference	652
13.171.1 Macro Definition Documentation	653
13.171.1.1 DNS_BIN_VER	653
13.171.1.2 mask_t	653
13.172 sell.h File Reference	653
13.173 sell_pspmv.inl File Reference	653
13.173.1 Macro Definition Documentation	654
13.173.1.1 __FFLASFFPACK_fflas_sparse_sell_pspmv_INL	654
13.174 sell_spmv.inl File Reference	654
13.174.1 Macro Definition Documentation	655
13.174.1.1 __FFLASFFPACK_fflas_sparse_sell_spmv_INL	655
13.175 sell_utils.inl File Reference	655
13.175.1 Macro Definition Documentation	655
13.175.1.1 __FFLASFFPACK_fflas_sparse_sell_utils_INL	655

13.176 sparse_matrix_traits.h File Reference . . . . .	656
13.177 utils.h File Reference . . . . .	657
13.178 fflas_transpose.h File Reference . . . . .	657
13.178.1 Detailed Description . . . . .	658
13.178.2 Macro Definition Documentation . . . . .	658
13.178.2.1 FFLAS_TRANSPOSE_BLOCKSIZE . . . . .	658
13.178.2.2 LD . . . . .	658
13.178.2.3 ST . . . . .	658
13.179 ffpack.doxy File Reference . . . . .	659
13.180 ffpack.h File Reference . . . . .	659
13.180.1 Detailed Description . . . . .	667
13.180.2 Macro Definition Documentation . . . . .	667
13.180.2.1 __FFLASFFPACK_FTRSTR_THRESHOLD . . . . .	667
13.180.2.2 __FFLASFFPACK_FTRSSYR2K_THRESHOLD . . . . .	667
13.181 ffpack.inl File Reference . . . . .	668
13.181.1 Macro Definition Documentation . . . . .	669
13.181.1.1 __FFLASFFPACK_ffpack_INL . . . . .	669
13.182 ffpack_bruhatgen.inl File Reference . . . . .	669
13.182.1 Macro Definition Documentation . . . . .	670
13.182.1.1 __FFLASFFPACK_ffpack_bruhatgen_inl . . . . .	670
13.183 ffpack_charpoly.inl File Reference . . . . .	670
13.183.1 Macro Definition Documentation . . . . .	671
13.183.1.1 __FFLASFFPACK_charpoly_INL . . . . .	671
13.184 ffpack_charpoly_danilevski.inl File Reference . . . . .	671
13.184.1 Macro Definition Documentation . . . . .	671
13.184.1.1 __FFLASFFPACK_ffpack_charpoly_danilveski_INL . . . . .	671
13.185 ffpack_charpoly_kgfast.inl File Reference . . . . .	671
13.185.1 Macro Definition Documentation . . . . .	672
13.185.1.1 __FFLASFFPACK_ffpack_charpoly_kgfast_INL . . . . .	672
13.186 ffpack_charpoly_kgfastgeneralized.inl File Reference . . . . .	672
13.186.1 Macro Definition Documentation . . . . .	672
13.186.1.1 __FFLASFFPACK_ffpack_charpoly_kgfastgeneralized_INL . . . . .	672
13.187 ffpack_charpoly_kglu.inl File Reference . . . . .	672
13.187.1 Macro Definition Documentation . . . . .	673
13.187.1.1 __FFLASFFPACK_ffpack_charpoly_kglu_INL . . . . .	673
13.188 ffpack_charpoly_mp.inl File Reference . . . . .	673
13.188.1 Macro Definition Documentation . . . . .	673
13.188.1.1 __FFPACK_charpoly_mp_INL . . . . .	673
13.189 ffpack_det_mp.inl File Reference . . . . .	673
13.189.1 Macro Definition Documentation . . . . .	674
13.189.1.1 __FFPACK_det_mp_INL . . . . .	674
13.190 ffpack_echelonforms.inl File Reference . . . . .	674

13.190.1 Macro Definition Documentation	675
13.190.1.1 __FFLASFFPACK_ffpack_echelon_forms_INL	675
13.190.1.2 __FFLASFFPACK_GAUSSJORDAN_BASECASE	675
13.191 fpack_fgesv.inl File Reference	675
13.191.1 Macro Definition Documentation	676
13.191.1.1 __FFLASFFPACK_ffpack_fgesv_INL	676
13.192 fpack_fgetrs.inl File Reference	676
13.192.1 Macro Definition Documentation	676
13.192.1.1 __FFLASFFPACK_ffpack_fgetrs_INL	676
13.193 fpack_frobenius.inl File Reference	676
13.194 fpack_fsytrf.inl File Reference	677
13.194.1 Macro Definition Documentation	678
13.194.1.1 __FFLASFFPACK_ffpack_fsytrf_INL	678
13.195 fpack_ftrssyr2k.inl File Reference	678
13.195.1 Macro Definition Documentation	679
13.195.1.1 __FFLASFFPACK_ffpack_ftrssyr2k_INL	679
13.196 fpack_ftrstr.inl File Reference	679
13.196.1 Macro Definition Documentation	679
13.196.1.1 __FFLASFFPACK_ffpack_ftrstr_INL	679
13.197 fpack_ftrtr.inl File Reference	679
13.197.1 Macro Definition Documentation	680
13.197.1.1 ENABLE_ALL_CHECKINGS	680
13.197.1.2 __FFLASFFPACK_ffpack_ftrtr_INL	680
13.198 fpack_invert.inl File Reference	680
13.198.1 Macro Definition Documentation	680
13.198.1.1 __FFLASFFPACK_ffpack_invert_INL	680
13.199 fpack_krylovelim.inl File Reference	681
13.199.1 Macro Definition Documentation	681
13.199.1.1 __FFLASFFPACK_ffpack_krylovelim_INL	681
13.200 fpack_ludivine.inl File Reference	681
13.200.1 Macro Definition Documentation	681
13.200.1.1 __FFLASFFPACK_ffpack_ludivine_INL	681
13.201 fpack_ludivine_mp.inl File Reference	682
13.201.1 Macro Definition Documentation	682
13.201.1.1 __FFPACK_ludivine_mp_INL	682
13.202 fpack_minpoly.inl File Reference	682
13.202.1 Macro Definition Documentation	683
13.202.1.1 __FFLASFFPACK_ffpack_minpoly_INL	683
13.203 fpack_permutation.inl File Reference	683
13.203.1 Macro Definition Documentation	685
13.203.1.1 __FFLASFFPACK_ffpack_permutation_INL	685
13.203.1.2 FFLASFFPACK_PERM_BKSIZE	685

13.204 ffpack_pluq.inl File Reference	685
13.204.1 Macro Definition Documentation	686
13.204.1.1 __FFLASFFPACK_ffpack_pluq_INL	686
13.204.1.2 CROUT	686
13.205 ffpack_pluq_mp.inl File Reference	686
13.205.1 Macro Definition Documentation	686
13.205.1.1 __FFPACK_pluq_mp_INL	686
13.206 ffpack_ppluq.inl File Reference	686
13.206.1 Macro Definition Documentation	687
13.206.1.1 __FFLASFFPACK_ffpack_ppluq_INL	687
13.206.1.2 __FFLAS__TRSM_READONLY	687
13.206.1.3 PBASECASE_K	687
13.207 ffpack_rankprofiles.inl File Reference	687
13.207.1 Macro Definition Documentation	688
13.207.1.1 __FFLASFFPACK_ffpack_rank_profiles_INL	688
13.208 field-traits.h File Reference	688
13.208.1 Detailed Description	691
13.209 field.doxy File Reference	691
13.210 rns-double-elt.h File Reference	691
13.210.1 Detailed Description	691
13.211 rns-double-recint.inl File Reference	691
13.211.1 Macro Definition Documentation	692
13.211.1.1 __FFLASFFPACK_field_rns_double_recint_INL	692
13.212 rns-double.h File Reference	692
13.212.1 Detailed Description	692
13.212.2 Macro Definition Documentation	692
13.212.2.1 ROUND_DOWN	692
13.213 rns-double.inl File Reference	693
13.213.1 Macro Definition Documentation	693
13.213.1.1 __FFLASFFPACK_field_rns_double_INL	693
13.214 rns-integer-mod.h File Reference	693
13.214.1 Detailed Description	694
13.215 rns-integer.h File Reference	694
13.215.1 Detailed Description	694
13.216 rns.h File Reference	694
13.217 rns.inl File Reference	695
13.217.1 Macro Definition Documentation	695
13.217.1.1 __FFLASFFPACK_field_rns_INL	695
13.218 interfaces.doxy File Reference	695
13.219 fflas_c.h File Reference	695
13.219.1 Macro Definition Documentation	697
13.219.1.1 FFLAS_COMPILED	697



13.219.2 Enumeration Type Documentation	697
13.219.2.1 FFLAS_C_ORDER	697
13.219.2.2 FFLAS_C_TRANSPOSE	697
13.219.2.3 FFLAS_C_UPLO	697
13.219.2.4 FFLAS_C_DIAG	698
13.219.2.5 FFLAS_C_SIDE	698
13.219.2.6 FFLAS_C_BASE	698
13.219.3 Function Documentation	698
13.219.3.1 freducein_1_modular_double()	698
13.219.3.2 freduce_1_modular_double()	699
13.219.3.3 fnegin_1_modular_double()	699
13.219.3.4 fneg_1_modular_double()	699
13.219.3.5 fzero_1_modular_double()	699
13.219.3.6 fiszero_1_modular_double()	699
13.219.3.7 fequal_1_modular_double()	699
13.219.3.8 fassign_1_modular_double()	700
13.219.3.9 fscal_1_modular_double()	700
13.219.3.10 fscal_1_modular_double()	700
13.219.3.11 faxpy_1_modular_double()	700
13.219.3.12 fdot_1_modular_double()	700
13.219.3.13 fswap_1_modular_double()	700
13.219.3.14 fadd_1_modular_double()	701
13.219.3.15 fsub_1_modular_double()	701
13.219.3.16 faddin_1_modular_double()	701
13.219.3.17 fsubin_1_modular_double()	701
13.219.3.18 fassign_2_modular_double()	701
13.219.3.19 fzero_2_modular_double()	702
13.219.3.20 fequal_2_modular_double()	702
13.219.3.21 fiszero_2_modular_double()	702
13.219.3.22 fidentity_2_modular_double()	702
13.219.3.23 freducein_2_modular_double()	702
13.219.3.24 freduce_2_modular_double()	703
13.219.3.25 fnegin_2_modular_double()	703
13.219.3.26 fneg_2_modular_double()	703
13.219.3.27 fscal_2_modular_double()	703
13.219.3.28 fscal_2_modular_double()	703
13.219.3.29 faxpy_2_modular_double()	704
13.219.3.30 fmove_2_modular_double()	704
13.219.3.31 fadd_2_modular_double()	704
13.219.3.32 fsub_2_modular_double()	704
13.219.3.33 fsubin_2_modular_double()	704
13.219.3.34 faddin_2_modular_double()	705

13.219.3.35 fgemv_2_modular_double()	705
13.219.3.36 fger_2_modular_double()	705
13.219.3.37 ftrsv_2_modular_double()	705
13.219.3.38 ftrsm_3_modular_double()	706
13.219.3.39 ftrmm_3_modular_double()	706
13.219.3.40 fgemm_3_modular_double()	706
13.219.3.41 fsquare_3_modular_double()	707
13.220 fflas_L1_inst.C File Reference	707
13.220.1 Macro Definition Documentation	707
13.220.1.1 __FFLAS_L1_INST_C	707
13.220.1.2 INST_OR_DECL	707
13.220.1.3 FFLAS_FIELD [1/2]	707
13.220.1.4 FFLAS_ELT [1/6]	707
13.220.1.5 FFLAS_ELT [2/6]	708
13.220.1.6 FFLAS_ELT [3/6]	708
13.220.1.7 FFLAS_FIELD [2/2]	708
13.220.1.8 FFLAS_ELT [4/6]	708
13.220.1.9 FFLAS_ELT [5/6]	708
13.220.1.10 FFLAS_ELT [6/6]	708
13.221 fflas_L1_inst.h File Reference	708
13.221.1 Macro Definition Documentation	708
13.221.1.1 INST_OR_DECL	708
13.221.1.2 FFLAS_FIELD [1/2]	708
13.221.1.3 FFLAS_ELT [1/6]	708
13.221.1.4 FFLAS_ELT [2/6]	709
13.221.1.5 FFLAS_ELT [3/6]	709
13.221.1.6 FFLAS_FIELD [2/2]	709
13.221.1.7 FFLAS_ELT [4/6]	709
13.221.1.8 FFLAS_ELT [5/6]	709
13.221.1.9 FFLAS_ELT [6/6]	709
13.222 fflas_L1_inst_implem.inl File Reference	709
13.223 fflas_L2_inst.C File Reference	710
13.223.1 Macro Definition Documentation	711
13.223.1.1 __FFLAS_L2_INST_C	711
13.223.1.2 INST_OR_DECL	711
13.223.1.3 FFLAS_FIELD [1/2]	711
13.223.1.4 FFLAS_ELT [1/6]	711
13.223.1.5 FFLAS_ELT [2/6]	711
13.223.1.6 FFLAS_ELT [3/6]	711
13.223.1.7 FFLAS_FIELD [2/2]	711
13.223.1.8 FFLAS_ELT [4/6]	711
13.223.1.9 FFLAS_ELT [5/6]	711

13.223.1.10 FFLAS_ELT [6/6]	711
13.224 fflas_L2_inst.h File Reference	711
13.224.1 Macro Definition Documentation	712
13.224.1.1 INST_OR_DECL	712
13.224.1.2 FFLAS_FIELD [1/2]	712
13.224.1.3 FFLAS_ELT [1/6]	712
13.224.1.4 FFLAS_ELT [2/6]	712
13.224.1.5 FFLAS_ELT [3/6]	712
13.224.1.6 FFLAS_FIELD [2/2]	712
13.224.1.7 FFLAS_ELT [4/6]	712
13.224.1.8 FFLAS_ELT [5/6]	712
13.224.1.9 FFLAS_ELT [6/6]	712
13.225 fflas_L2_inst_implem.inl File Reference	712
13.226 fflas_L3_inst.C File Reference	714
13.226.1 Macro Definition Documentation	714
13.226.1.1 __FFLAS_L3_INST_C	714
13.226.1.2 INST_OR_DECL	714
13.226.1.3 FFLAS_FIELD [1/2]	714
13.226.1.4 FFLAS_ELT [1/6]	714
13.226.1.5 FFLAS_ELT [2/6]	715
13.226.1.6 FFLAS_ELT [3/6]	715
13.226.1.7 FFLAS_FIELD [2/2]	715
13.226.1.8 FFLAS_ELT [4/6]	715
13.226.1.9 FFLAS_ELT [5/6]	715
13.226.1.10 FFLAS_ELT [6/6]	715
13.227 fflas_L3_inst.h File Reference	715
13.227.1 Macro Definition Documentation	715
13.227.1.1 INST_OR_DECL	715
13.227.1.2 FFLAS_FIELD [1/2]	715
13.227.1.3 FFLAS_ELT [1/6]	715
13.227.1.4 FFLAS_ELT [2/6]	716
13.227.1.5 FFLAS_ELT [3/6]	716
13.227.1.6 FFLAS_FIELD [2/2]	716
13.227.1.7 FFLAS_ELT [4/6]	716
13.227.1.8 FFLAS_ELT [5/6]	716
13.227.1.9 FFLAS_ELT [6/6]	716
13.228 fflas_L3_inst_implem.inl File Reference	716
13.228.1 Macro Definition Documentation	717
13.228.1.1 __FFLAS__TRSM_READONLY	717
13.229 fflas_lvl1.C File Reference	717
13.229.1 Detailed Description	718
13.229.2 Function Documentation	718

13.229.2.1 freducein_1_modular_double()	718
13.229.2.2 freduce_1_modular_double()	718
13.229.2.3 fnegin_1_modular_double()	718
13.229.2.4 fneg_1_modular_double()	718
13.229.2.5 fzero_1_modular_double()	719
13.229.2.6 fiszero_1_modular_double()	719
13.229.2.7 fequal_1_modular_double()	719
13.229.2.8 fassign_1_modular_double()	719
13.229.2.9 fscaln_1_modular_double()	719
13.229.2.10 fscal_1_modular_double()	719
13.229.2.11 faxpy_1_modular_double()	720
13.229.2.12 fdot_1_modular_double()	720
13.229.2.13 fswap_1_modular_double()	720
13.229.2.14 fadd_1_modular_double()	720
13.229.2.15 fsub_1_modular_double()	720
13.229.2.16 faddin_1_modular_double()	721
13.229.2.17 fsubin_1_modular_double()	721
13.230 fflas_lvl2.C File Reference	721
13.230.1 Detailed Description	722
13.230.2 Function Documentation	722
13.230.2.1 fassign_2_modular_double()	722
13.230.2.2 fzero_2_modular_double()	722
13.230.2.3 fequal_2_modular_double()	723
13.230.2.4 fiszero_2_modular_double()	723
13.230.2.5 fidentity_2_modular_double()	723
13.230.2.6 freducein_2_modular_double()	723
13.230.2.7 freduce_2_modular_double()	723
13.230.2.8 fnegin_2_modular_double()	723
13.230.2.9 fneg_2_modular_double()	724
13.230.2.10 fscaln_2_modular_double()	724
13.230.2.11 fscal_2_modular_double()	724
13.230.2.12 faxpy_2_modular_double()	724
13.230.2.13 fmove_2_modular_double()	724
13.230.2.14 fadd_2_modular_double()	725
13.230.2.15 fsub_2_modular_double()	725
13.230.2.16 fsubin_2_modular_double()	725
13.230.2.17 faddin_2_modular_double()	725
13.230.2.18 fgemv_2_modular_double()	726
13.230.2.19 fger_2_modular_double()	726
13.230.2.20 ftrsv_2_modular_double()	726
13.231 fflas_lvl3.C File Reference	726
13.231.1 Detailed Description	727

13.231.2 Function Documentation	727
13.231.2.1 ftrsm_3_modular_double()	727
13.231.2.2 ftrmm_3_modular_double()	727
13.231.2.3 fgemm_3_modular_double()	728
13.231.2.4 fsquare_3_modular_double()	728
13.232 fflas_sparse.C File Reference	728
13.232.1 Detailed Description	728
13.233 ffpack.C File Reference	728
13.233.1 Detailed Description	732
13.233.2 Function Documentation	732
13.233.2.1 LAPACKPerm2MathPerm()	732
13.233.2.2 MathPerm2LAPACKPerm()	732
13.233.2.3 MatrixApplyS_modular_double()	732
13.233.2.4 PermApplyS_double()	732
13.233.2.5 MatrixApplyT_modular_double()	733
13.233.2.6 PermApplyT_double()	733
13.233.2.7 composePermutationsLLM()	733
13.233.2.8 composePermutationsLLL()	733
13.233.2.9 composePermutationsMLM()	733
13.233.2.10 cyclic_shift_mathPerm()	733
13.233.2.11 cyclic_shift_row_modular_double()	734
13.233.2.12 cyclic_shift_col_modular_double()	734
13.233.2.13 applyP_modular_double()	734
13.233.2.14 fgetrsin_modular_double()	734
13.233.2.15 fgetrsv_modular_double()	734
13.233.2.16 fgesvin_modular_double()	735
13.233.2.17 fgesv_modular_double()	735
13.233.2.18 ftrtri_modular_double()	735
13.233.2.19 trinv_left_modular_double()	735
13.233.2.20 ftrtrm_modular_double()	736
13.233.2.21 PLUQ_modular_double()	736
13.233.2.22 LUdivine_modular_double()	736
13.233.2.23 ColumnEchelonForm_modular_double()	736
13.233.2.24 RowEchelonForm_modular_double()	737
13.233.2.25 ReducedColumnEchelonForm_modular_double()	737
13.233.2.26 ReducedRowEchelonForm_modular_double()	737
13.233.2.27 ColumnEchelonForm_modular_float()	737
13.233.2.28 RowEchelonForm_modular_float()	737
13.233.2.29 ReducedColumnEchelonForm_modular_float()	738
13.233.2.30 ReducedRowEchelonForm_modular_float()	738
13.233.2.31 ColumnEchelonForm_modular_int32_t()	738
13.233.2.32 RowEchelonForm_modular_int32_t()	738

13.233.2.33 ReducedColumnEchelonForm_modular_int32_t()	739
13.233.2.34 ReducedRowEchelonForm_modular_int32_t()	739
13.233.2.35 pColumnEchelonForm_modular_double()	739
13.233.2.36 pRowEchelonForm_modular_double()	739
13.233.2.37 pReducedColumnEchelonForm_modular_double()	740
13.233.2.38 pReducedRowEchelonForm_modular_double()	740
13.233.2.39 pColumnEchelonForm_modular_float()	740
13.233.2.40 pRowEchelonForm_modular_float()	740
13.233.2.41 pReducedColumnEchelonForm_modular_float()	741
13.233.2.42 pReducedRowEchelonForm_modular_float()	741
13.233.2.43 pColumnEchelonForm_modular_int32_t()	741
13.233.2.44 pRowEchelonForm_modular_int32_t()	741
13.233.2.45 pReducedColumnEchelonForm_modular_int32_t()	741
13.233.2.46 pReducedRowEchelonForm_modular_int32_t()	742
13.233.2.47 Invertin_modular_double()	742
13.233.2.48 Invert_modular_double()	742
13.233.2.49 Invert2_modular_double()	742
13.233.2.50 KrylovElim_modular_double()	743
13.233.2.51 SpecRankProfile_modular_double()	743
13.233.2.52 Rank_modular_double()	743
13.233.2.53 IsSingular_modular_double()	743
13.233.2.54 Det_modular_double()	743
13.233.2.55 Solve_modular_double()	744
13.233.2.56 solveLB_modular_double()	744
13.233.2.57 solveLB2_modular_double()	744
13.233.2.58 RandomNullSpaceVector_modular_double()	744
13.233.2.59 NullSpaceBasis_modular_double()	744
13.233.2.60 RowRankProfile_modular_double()	745
13.233.2.61 ColumnRankProfile_modular_double()	745
13.233.2.62 RankProfileFromLU()	745
13.233.2.63 LeadingSubmatrixRankProfiles()	745
13.233.2.64 RowRankProfileSubmatrixIndices_modular_double()	745
13.233.2.65 ColRankProfileSubmatrixIndices_modular_double()	746
13.233.2.66 RowRankProfileSubmatrix_modular_double()	746
13.233.2.67 ColRankProfileSubmatrix_modular_double()	746
13.233.2.68 getTriangular_modular_double()	746
13.233.2.69 getTriangularin_modular_double()	747
13.233.2.70 getEchelonForm_modular_double()	747
13.233.2.71 getEchelonFormin_modular_double()	747
13.233.2.72 getEchelonTransform_modular_double()	747
13.233.2.73 getReducedEchelonForm_modular_double()	748
13.233.2.74 getReducedEchelonFormin_modular_double()	748

13.233.2.75 getReducedEchelonTransform_modular_double()	748
13.233.2.76 PLUQtoEchelonPermutation()	749
13.234 fpack_c.h File Reference	749
13.234.1 Macro Definition Documentation	752
13.234.1.1 FFPACK_COMPILED	752
13.234.2 Enumeration Type Documentation	752
13.234.2.1 FFLAS_C_ORDER	752
13.234.2.2 FFLAS_C_TRANSPOSE	752
13.234.2.3 FFLAS_C_UPLO	752
13.234.2.4 FFLAS_C_DIAG	752
13.234.2.5 FFLAS_C_SIDE	753
13.234.2.6 FFPACK_C_LU_TAG	753
13.234.2.7 FFPACK_C_CHARPOLY_TAG	753
13.234.2.8 FFPACK_C_MINPOLY_TAG	753
13.234.3 Function Documentation	753
13.234.3.1 LAPACKPerm2MathPerm()	753
13.234.3.2 MathPerm2LAPACKPerm()	754
13.234.3.3 MatrixApplyS_modular_double()	754
13.234.3.4 PermApplyS_double()	754
13.234.3.5 MatrixApplyT_modular_double()	754
13.234.3.6 PermApplyT_double()	754
13.234.3.7 composePermutationsLLM()	755
13.234.3.8 composePermutationsLLL()	755
13.234.3.9 composePermutationsMLM()	755
13.234.3.10 cyclic_shift_mathPerm()	755
13.234.3.11 cyclic_shift_row_modular_double()	755
13.234.3.12 cyclic_shift_col_modular_double()	755
13.234.3.13 applyP_modular_double()	755
13.234.3.14 fgetrsin_modular_double()	756
13.234.3.15 fgetrs_modular_double()	756
13.234.3.16 fgesvin_modular_double()	756
13.234.3.17 fgesv_modular_double()	756
13.234.3.18 ftrtri_modular_double()	757
13.234.3.19 trinv_left_modular_double()	757
13.234.3.20 ftrtrm_modular_double()	757
13.234.3.21 PLUQ_modular_double()	757
13.234.3.22 LUdivine_modular_double()	758
13.234.3.23 LUdivine_small_modular_double()	758
13.234.3.24 LUdivine_gauss_modular_double()	758
13.234.3.25 ColumnEchelonForm_modular_double()	758
13.234.3.26 RowEchelonForm_modular_double()	759
13.234.3.27 ColumnEchelonForm_modular_float()	759

13.234.3.28 RowEchelonForm_modular_float()	759
13.234.3.29 ColumnEchelonForm_modular_int32_t()	759
13.234.3.30 RowEchelonForm_modular_int32_t()	759
13.234.3.31 ReducedColumnEchelonForm_modular_double()	760
13.234.3.32 ReducedRowEchelonForm_modular_double()	760
13.234.3.33 ReducedColumnEchelonForm_modular_float()	760
13.234.3.34 ReducedRowEchelonForm_modular_float()	760
13.234.3.35 ReducedColumnEchelonForm_modular_int32_t()	761
13.234.3.36 ReducedRowEchelonForm_modular_int32_t()	761
13.234.3.37 ReducedRowEchelonForm2_modular_double()	761
13.234.3.38 REF_modular_double()	761
13.234.3.39 Invertin_modular_double()	762
13.234.3.40 Invert_modular_double()	762
13.234.3.41 Invert2_modular_double()	762
13.234.3.42 KrylovElim_modular_double()	762
13.234.3.43 SpecRankProfile_modular_double()	762
13.234.3.44 Rank_modular_double()	763
13.234.3.45 IsSingular_modular_double()	763
13.234.3.46 Det_modular_double()	763
13.234.3.47 Solve_modular_double()	763
13.234.3.48 solveLB_modular_double()	763
13.234.3.49 solveLB2_modular_double()	764
13.234.3.50 RandomNullSpaceVector_modular_double()	764
13.234.3.51 NullSpaceBasis_modular_double()	764
13.234.3.52 RowRankProfile_modular_double()	764
13.234.3.53 ColumnRankProfile_modular_double()	765
13.234.3.54 RankProfileFromLU()	765
13.234.3.55 LeadingSubmatrixRankProfiles()	765
13.234.3.56 RowRankProfileSubmatrixIndices_modular_double()	765
13.234.3.57 ColRankProfileSubmatrixIndices_modular_double()	765
13.234.3.58 RowRankProfileSubmatrix_modular_double()	766
13.234.3.59 ColRankProfileSubmatrix_modular_double()	766
13.234.3.60 getTriangular_modular_double()	766
13.234.3.61 getTriangularin_modular_double()	766
13.234.3.62 getEchelonForm_modular_double()	766
13.234.3.63 getEchelonFormin_modular_double()	767
13.234.3.64 getEchelonTransform_modular_double()	767
13.234.3.65 getReducedEchelonForm_modular_double()	767
13.234.3.66 getReducedEchelonFormin_modular_double()	768
13.234.3.67 getReducedEchelonTransform_modular_double()	768
13.234.3.68 PLUQtoEchelonPermutation()	768
13.235 ffpack_inst.C File Reference	768



13.235.1 Macro Definition Documentation	769
13.235.1.1 __FFPACK_INST_C	769
13.235.1.2 FFLAS_COMPILED	769
13.235.1.3 INST_OR_DECL	769
13.235.1.4 FFLAS_FIELD [1/2]	769
13.235.1.5 FFLAS_ELT [1/6]	769
13.235.1.6 FFLAS_ELT [2/6]	769
13.235.1.7 FFLAS_ELT [3/6]	769
13.235.1.8 FFLAS_FIELD [2/2]	769
13.235.1.9 FFLAS_ELT [4/6]	769
13.235.1.10 FFLAS_ELT [5/6]	769
13.235.1.11 FFLAS_ELT [6/6]	769
13.236 fpack_inst.h File Reference	769
13.236.1 Macro Definition Documentation	770
13.236.1.1 FFLAS_COMPILED	770
13.236.1.2 INST_OR_DECL	770
13.236.1.3 FFLAS_FIELD [1/2]	770
13.236.1.4 FFLAS_ELT [1/6]	770
13.236.1.5 FFLAS_ELT [2/6]	770
13.236.1.6 FFLAS_ELT [3/6]	770
13.236.1.7 FFLAS_FIELD [2/2]	770
13.236.1.8 FFLAS_ELT [4/6]	770
13.236.1.9 FFLAS_ELT [5/6]	770
13.236.1.10 FFLAS_ELT [6/6]	770
13.237 fpack_inst_implem.inl File Reference	771
13.238 blockcuts.inl File Reference	774
13.238.1 Macro Definition Documentation	775
13.238.1.1 __FFLASFFPACK_fflas_blockcuts_INL	775
13.238.1.2 __FFLASFFPACK_MINBLOCKCUTS	775
13.239 fflas_plevel1.h File Reference	775
13.240 kaapi_routines.inl File Reference	776
13.240.1 Macro Definition Documentation	776
13.240.1.1 __FFLASFFPACK_KAAPI_ROUTINES_INL	776
13.241 parallel.h File Reference	776
13.241.1 Macro Definition Documentation	777
13.241.1.1 __FFLASFFPACK_SEQUENTIAL	777
13.241.1.2 index_t	777
13.241.1.3 TASK	777
13.241.1.4 WAIT	777
13.241.1.5 CHECK_DEPENDENCIES	777
13.241.1.6 BARRIER	777
13.241.1.7 PAR_BLOCK	777

13.241.1.8 SYNCH_GROUP . . . . .	777
13.241.1.9 THREAD_INDEX . . . . .	778
13.241.1.10 NUM_THREADS . . . . .	778
13.241.1.11 SET_THREADS . . . . .	778
13.241.1.12 MAX_THREADS . . . . .	778
13.241.1.13 READ . . . . .	778
13.241.1.14 WRITE . . . . .	778
13.241.1.15 READWRITE . . . . .	778
13.241.1.16 CONSTREFERENCE . . . . .	778
13.241.1.17 VALUE . . . . .	778
13.241.1.18 BEGIN_PARALLEL_MAIN . . . . .	778
13.241.1.19 END_PARALLEL_MAIN . . . . .	778
13.241.1.20 FORBLOCK1D . . . . .	779
13.241.1.21 FOR1D . . . . .	779
13.241.1.22 PARFORBLOCK1D . . . . .	779
13.241.1.23 PARFOR1D . . . . .	779
13.241.1.24 FORBLOCK2D . . . . .	779
13.241.1.25 FOR2D . . . . .	780
13.241.1.26 PARFORBLOCK2D . . . . .	780
13.241.1.27 PARFOR2D . . . . .	780
13.241.1.28 COMMA . . . . .	780
13.241.1.29 MODE . . . . .	780
13.241.1.30 RETURNPARAM . . . . .	780
13.241.1.31 NUMARGS . . . . .	781
13.241.1.32 PP_NARG_ . . . . .	781
13.241.1.33 PP_ARG_N . . . . .	781
13.241.1.34 PP_RSEQ_N . . . . .	782
13.241.1.35 NOSPLIT . . . . .	782
13.241.1.36 splitting_0 . . . . .	782
13.241.1.37 splitting_1 . . . . .	782
13.241.1.38 splitting_2 . . . . .	782
13.241.1.39 splitting_3 . . . . .	783
13.241.1.40 splitt . . . . .	783
13.241.1.41 SPLITTER . . . . .	783
13.242 pfgemm_variants.inl File Reference . . . . .	783
13.243 pfgemv.inl File Reference . . . . .	784
13.244 align-allocator.h File Reference . . . . .	784
13.245 args-parser.h File Reference . . . . .	784
13.245.1 Macro Definition Documentation . . . . .	785
13.245.1.1 TYPE_BOOL . . . . .	785
13.245.1.2 END_OF_ARGUMENTS . . . . .	785
13.245.1.3 type_integer . . . . .	785

13.245.2 Enumeration Type Documentation	785
13.245.2.1 ArgumentType	785
13.245.3 Function Documentation	786
13.245.3.1 printHelpMessage()	786
13.245.3.2 findArgument()	786
13.245.3.3 getListArgs()	786
13.246 bit_manipulation.h File Reference	786
13.246.1 Macro Definition Documentation	787
13.246.1.1 __has_builtin	787
13.246.2 Function Documentation	787
13.246.2.1 clz() [1/2]	787
13.246.2.2 clz() [2/2]	787
13.246.2.3 ctz() [1/2]	787
13.246.2.4 ctz() [2/2]	787
13.247 cast.h File Reference	787
13.248 debug.h File Reference	787
13.248.1 Detailed Description	788
13.248.2 Macro Definition Documentation	788
13.248.2.1 FFLASFFPACK_check	788
13.248.2.2 FFLASFFPACK_abort	788
13.249 fflas_intrinsic.h File Reference	788
13.250 fflas_io.h File Reference	788
13.251 fflas_memory.h File Reference	789
13.252 fflas_randommatrix.h File Reference	790
13.253 flimits.h File Reference	792
13.253.1 Function Documentation	792
13.253.1.1 in_range() [1/3]	792
13.253.1.2 in_range() [2/3]	793
13.253.1.3 in_range() [3/3]	793
13.254 Matio.h File Reference	793
13.254.1 Function Documentation	793
13.254.1.1 read_field()	793
13.254.1.2 write_field()	793
13.255 test-utils.h File Reference	793
13.256 timer.h File Reference	794
13.257 cblas.C File Reference	794
13.257.1 Macro Definition Documentation	795
13.257.1.1 __FFLASFFPACK_CONFIGURATION	795
13.257.1.2 __FFLASFFPACK_HAVE_CBLAS	795
13.257.2 Function Documentation	795
13.257.2.1 main()	795
13.258 clapack.C File Reference	795

13.258.1 Macro Definition Documentation . . . . .	795
13.258.1.1 __FFLASFFPACK_CONFIGURATION . . . . .	795
13.258.1.2 __FFLASFFPACK_HAVE_LAPACK . . . . .	795
13.258.1.3 __FFLASFFPACK_HAVE_CLAPACK . . . . .	795
13.258.2 Function Documentation . . . . .	795
13.258.2.1 main() . . . . .	795
13.259 cuda.C File Reference . . . . .	796
13.259.1 Function Documentation . . . . .	796
13.259.1.1 main() . . . . .	796
13.260 fblas.C File Reference . . . . .	796
13.260.1 Macro Definition Documentation . . . . .	796
13.260.1.1 __FFLASFFPACK_CONFIGURATION . . . . .	796
13.260.2 Function Documentation . . . . .	796
13.260.2.1 dgemm_() . . . . .	796
13.260.2.2 main() . . . . .	797
13.261 lapack.C File Reference . . . . .	797
13.261.1 Macro Definition Documentation . . . . .	797
13.261.1.1 __FFLASFFPACK_CONFIGURATION . . . . .	797
13.261.1.2 __FFLASFFPACK_HAVE_LAPACK . . . . .	797
13.261.2 Function Documentation . . . . .	797
13.261.2.1 main() . . . . .	797
13.262 regression-check.C File Reference . . . . .	797
13.262.1 Function Documentation . . . . .	797
13.262.1.1 check1() . . . . .	797
13.262.1.2 check2() . . . . .	798
13.262.1.3 check3() . . . . .	798
13.262.1.4 check4() . . . . .	798
13.262.1.5 checkZeroDimCharpoly() . . . . .	798
13.262.1.6 checkZeroDimMinPoly() . . . . .	798
13.262.1.7 gf2ModularBalanced() . . . . .	798
13.262.1.8 main() . . . . .	798
13.263 test-charpoly-check.C File Reference . . . . .	798
13.263.1 Macro Definition Documentation . . . . .	798
13.263.1.1 ENABLE_CHECKER_charpoly . . . . .	798
13.263.1.2 TIME_CHECKER_CHARPOLY . . . . .	798
13.263.2 Function Documentation . . . . .	799
13.263.2.1 printPolynomial() . . . . .	799
13.263.2.2 main() . . . . .	799
13.264 test-charpoly.C File Reference . . . . .	799
13.264.1 Function Documentation . . . . .	799
13.264.1.1 launch_test() . . . . .	799
13.264.1.2 run_with_field() . . . . .	799

13.264.1.3 main()	800
13.265 test-compressQ.C File Reference	800
13.265.1 Typedef Documentation	800
13.265.1.1 Field	800
13.265.2 Function Documentation	800
13.265.2.1 printvect()	800
13.265.2.2 main()	800
13.266 test-det-check.C File Reference	801
13.266.1 Macro Definition Documentation	801
13.266.1.1 ENABLE_CHECKER_Det	801
13.266.1.2 TIME_CHECKER_Det	801
13.266.2 Function Documentation	801
13.266.2.1 main()	801
13.267 test-det.C File Reference	801
13.267.1 Function Documentation	802
13.267.1.1 test_det()	802
13.267.1.2 main()	802
13.268 test-echelon.C File Reference	802
13.268.1 Macro Definition Documentation	803
13.268.1.1 __FFLASFFPACK_SEQUENTIAL	803
13.268.1.2 __FFLASFFPACK_GAUSSJORDAN_BASECASE	803
13.268.1.3 __FFLASFFPACK_PLUQ_THRESHOLD	803
13.268.2 Function Documentation	803
13.268.2.1 test_colechelon()	803
13.268.2.2 test_rowechelon()	803
13.268.2.3 test_redcolechelon()	803
13.268.2.4 test_redrowechelon()	804
13.268.2.5 run_with_field()	804
13.268.2.6 main()	804
13.269 test-fadd.C File Reference	804
13.269.1 Function Documentation	805
13.269.1.1 test_fadd()	805
13.269.1.2 test_faddin()	805
13.269.1.3 test_fsub()	805
13.269.1.4 test_fsubin()	805
13.269.1.5 main()	805
13.270 test-fdot.C File Reference	805
13.270.1 Macro Definition Documentation	806
13.270.1.1 ENABLE_ALL_CHECKINGS	806
13.270.2 Function Documentation	806
13.270.2.1 check_fdot()	806
13.270.2.2 run_with_field()	806

13.270.2.3 run_with_Integer()	806
13.270.2.4 main()	807
13.271 test-fgemm-check.C File Reference	807
13.271.1 Macro Definition Documentation	807
13.271.1.1 ENABLE_ALL_CHECKINGS	807
13.271.2 Function Documentation	807
13.271.2.1 launch_MM_dispatch()	807
13.271.2.2 run_with_field()	808
13.271.2.3 main()	808
13.272 test-fgemm.C File Reference	808
13.272.1 Macro Definition Documentation	809
13.272.1.1 ENABLE_CHECKER_fgemm	809
13.272.2 Function Documentation	809
13.272.2.1 check_MM()	809
13.272.2.2 launch_MM()	809
13.272.2.3 launch_MM_dispatch()	809
13.272.2.4 run_with_field()	810
13.272.2.5 main()	810
13.273 test-fgemv.C File Reference	810
13.273.1 Function Documentation	811
13.273.1.1 check_MV()	811
13.273.1.2 launch_MV()	811
13.273.1.3 launch_MV_dispatch()	811
13.273.1.4 run_with_field()	812
13.273.1.5 main()	812
13.274 test-fger.C File Reference	812
13.274.1 Macro Definition Documentation	812
13.274.1.1 TIME	812
13.274.2 Function Documentation	813
13.274.2.1 check_fger()	813
13.274.2.2 launch_fger()	813
13.274.2.3 launch_fger_dispatch()	813
13.274.2.4 run_with_field()	813
13.274.2.5 main()	814
13.275 test-fgesv.C File Reference	814
13.275.1 Function Documentation	814
13.275.1.1 test_square_fgesv()	814
13.275.1.2 test_rect_fgesv()	814
13.275.1.3 run_with_field()	815
13.275.1.4 main()	815
13.276 test-finit.C File Reference	815
13.276.1 Function Documentation	815

13.276.1.1 test_freduce()	815
13.276.1.2 run_with_field()	816
13.276.1.3 main()	816
13.277 test-fscal.C File Reference	816
13.277.1 Function Documentation	816
13.277.1.1 test_fscal() [1/2]	816
13.277.1.2 test_fscal() [2/2]	817
13.277.1.3 test_fscaln() [1/2]	817
13.277.1.4 test_fscaln() [2/2]	817
13.277.1.5 main()	817
13.277.1.6 RandomMatrix()	817
13.278 test-fsyr2k.C File Reference	818
13.278.1 Macro Definition Documentation	818
13.278.1.1 ENABLE_ALL_CHECKINGS	818
13.278.2 Function Documentation	818
13.278.2.1 check_fsyr2k()	818
13.278.2.2 run_with_field()	819
13.278.2.3 main()	819
13.279 test-fsyrk.C File Reference	819
13.279.1 Macro Definition Documentation	820
13.279.1.1 ENABLE_ALL_CHECKINGS	820
13.279.2 Function Documentation	820
13.279.2.1 check_fsyrk()	820
13.279.2.2 check_fsyrk_diag()	820
13.279.2.3 check_fsyrk_bkdiag()	820
13.279.2.4 check_computeS1S2()	820
13.279.2.5 run_with_field()	821
13.279.2.6 main()	821
13.280 test-fsytrf.C File Reference	821
13.280.1 Function Documentation	821
13.280.1.1 operator<<()	821
13.280.1.2 test_RPM_fsytrf()	822
13.280.1.3 test_generic_fsytrf()	822
13.280.1.4 run_with_field()	822
13.280.1.5 main()	822
13.281 test-frm.C File Reference	822
13.281.1 Macro Definition Documentation	823
13.281.1.1 __FFLASFFPACK_SEQUENTIAL	823
13.281.2 Function Documentation	823
13.281.2.1 check_frm()	823
13.281.2.2 run_with_field()	823
13.281.2.3 main()	823

13.282 test-frmvc.C File Reference	823
13.282.1 Macro Definition Documentation	824
13.282.1.1 __FFLASFFPACK_SEQUENTIAL	824
13.282.1.2 ENABLE_ALL_CHECKINGS	824
13.282.2 Function Documentation	824
13.282.2.1 check_frmvc()	824
13.282.2.2 run_with_field()	824
13.282.2.3 main()	825
13.283 test-frm-check.C File Reference	825
13.283.1 Macro Definition Documentation	825
13.283.1.1 ENABLE_ALL_CHECKINGS	825
13.283.2 Function Documentation	825
13.283.2.1 main()	825
13.284 test-frm.C File Reference	825
13.284.1 Macro Definition Documentation	826
13.284.1.1 __FFLASFFPACK_SEQUENTIAL	826
13.284.1.2 ENABLE_ALL_CHECKINGS	826
13.284.2 Function Documentation	826
13.284.2.1 check_frm()	826
13.284.2.2 run_with_field()	826
13.284.2.3 main()	826
13.285 test-frmssyr2k.C File Reference	826
13.285.1 Macro Definition Documentation	827
13.285.1.1 ENABLE_ALL_CHECKINGS	827
13.285.2 Function Documentation	827
13.285.2.1 check_frmssyr2k()	827
13.285.2.2 run_with_field()	827
13.285.2.3 main()	827
13.286 test-frmstr.C File Reference	828
13.286.1 Macro Definition Documentation	828
13.286.1.1 ENABLE_ALL_CHECKINGS	828
13.286.2 Function Documentation	828
13.286.2.1 check_frmstr()	828
13.286.2.2 run_with_field()	828
13.286.2.3 main()	829
13.287 test-frmvc.C File Reference	829
13.287.1 Macro Definition Documentation	829
13.287.1.1 __FFLASFFPACK_SEQUENTIAL	829
13.287.1.2 ENABLE_ALL_CHECKINGS	829
13.287.2 Function Documentation	829
13.287.2.1 check_frmvc()	829
13.287.2.2 run_with_field()	830



13.287.2.3 main()	830
13.288 test-ftsrti.C File Reference	830
13.288.1 Macro Definition Documentation	830
13.288.1.1 __FFLASFFPACK_SEQUENTIAL	830
13.288.1.2 ENABLE_ALL_CHECKINGS	830
13.288.2 Function Documentation	830
13.288.2.1 check_ftsrti()	830
13.288.2.2 run_with_field()	831
13.288.2.3 main()	831
13.289 test-interfaces-c.c File Reference	831
13.289.1 Function Documentation	831
13.289.1.1 main()	831
13.290 test-invert-check.C File Reference	831
13.290.1 Macro Definition Documentation	832
13.290.1.1 ENABLE_ALL_CHECKINGS	832
13.290.2 Function Documentation	832
13.290.2.1 main()	832
13.291 test-io.C File Reference	832
13.291.1 Function Documentation	832
13.291.1.1 run_with_field()	832
13.291.1.2 main()	832
13.292 test-lu.C File Reference	833
13.292.1 Macro Definition Documentation	833
13.292.1.1 BASECASE_K	833
13.292.1.2 __FFLASFFPACK_SEQUENTIAL	834
13.292.1.3 __LUDIVINE_CUTOFF	834
13.292.2 Function Documentation	834
13.292.2.1 test_LUdivine()	834
13.292.2.2 verifPLUQ()	834
13.292.2.3 test_pluq()	835
13.292.2.4 launch_test()	836
13.292.2.5 run_with_field()	836
13.292.2.6 main()	836
13.292.3 Variable Documentation	836
13.292.3.1 tperm	836
13.292.3.2 tgemm	836
13.292.3.3 tBC	836
13.292.3.4 ttrsm	836
13.292.3.5 trest	836
13.292.3.6 timtot	836
13.292.3.7 mvcnt	836
13.293 test-maxdelayeddim.C File Reference	837

13.293.1 Macro Definition Documentation	837
13.293.1.1 MAX_WITH_SIZE_T	837
13.293.2 Function Documentation	837
13.293.2.1 test()	837
13.293.2.2 main()	837
13.294 test-minpoly.C File Reference	837
13.294.1 Function Documentation	838
13.294.1.1 check_minpoly()	838
13.294.1.2 run_with_field()	838
13.294.1.3 main()	838
13.295 test-multifile1.C File Reference	838
13.296 test-multifile2.C File Reference	838
13.296.1 Function Documentation	838
13.296.1.1 main()	838
13.297 test-nullspace.C File Reference	839
13.297.1 Function Documentation	839
13.297.1.1 checkingMessage()	839
13.297.1.2 readOrRandomMatrixWithRankAndRandomRPM()	839
13.297.1.3 test_nullspace()	839
13.297.1.4 run_with_field()	840
13.297.1.5 main()	840
13.298 test-permutations.C File Reference	840
13.298.1 Function Documentation	840
13.298.1.1 checkMonotonicApplyP()	840
13.298.1.2 main()	840
13.298.2 Variable Documentation	841
13.298.2.1 tperm	841
13.298.2.2 tgemm	841
13.298.2.3 tBC	841
13.298.2.4 ttrsm	841
13.298.2.5 trest	841
13.298.2.6 timtot	841
13.299 test-pluq-check.C File Reference	841
13.299.1 Macro Definition Documentation	841
13.299.1.1 ENABLE_ALL_CHECKINGS	841
13.299.2 Function Documentation	841
13.299.2.1 main()	841
13.300 test-quasisep.C File Reference	842
13.300.1 Function Documentation	842
13.300.1.1 test_BruhatGenerator()	842
13.300.1.2 launch_test()	842
13.300.1.3 testLTQSRPM()	842

13.300.1.4 run_with_field()	843
13.300.1.5 main()	843
13.301 test-rankprofiles.C File Reference	843
13.301.1 Macro Definition Documentation	843
13.301.1.1 __FFLASFFPACK_SEQUENTIAL	843
13.301.2 Function Documentation	843
13.301.2.1 run_with_field()	843
13.301.2.2 main()	844
13.302 test-rpm.C File Reference	844
13.302.1 Function Documentation	844
13.302.1.1 checkRPM()	844
13.302.1.2 checkSymmetricRPM()	844
13.302.1.3 main()	844
13.303 test-simd.C File Reference	844
13.303.1 Macro Definition Documentation	846
13.303.1.1 _TEST_ONE	846
13.303.1.2 TEST_ONE_OP	846
13.303.1.3 TEST_ONE_OP_WZ	846
13.303.1.4 TEST_IMPL	846
13.303.2 Function Documentation	847
13.303.2.1 check_eq() [1/2]	847
13.303.2.2 check_eq() [2/2]	847
13.303.2.3 cmp()	847
13.303.2.4 eval_func_on_array() [1/3]	847
13.303.2.5 eval_func_on_array() [2/3]	847
13.303.2.6 eval_func_on_array() [3/3]	847
13.303.2.7 operator<<()	847
13.303.2.8 test_impl_base() [1/2]	847
13.303.2.9 test_impl_base() [2/2]	848
13.303.2.10 test_impl()	848
13.303.2.11 main()	848
13.304 test-solve.C File Reference	848
13.304.1 Function Documentation	848
13.304.1.1 check_solve()	848
13.304.1.2 run_with_field()	848
13.304.1.3 main()	849
13.305 test-storage-transpose.C File Reference	849
13.305.1 Function Documentation	849
13.305.1.1 main()	849
13.306 101-fgemv.C File Reference	849
13.306.1 Function Documentation	849
13.306.1.1 main()	849

13.307 2x2-fgemv.C File Reference . . . . .	850
13.307.1 Function Documentation . . . . .	850
13.307.1.1 main() . . . . .	850
13.308 2x2-ftsrv.C File Reference . . . . .	850
13.308.1 Function Documentation . . . . .	850
13.308.1.1 main() . . . . .	850
13.309 2x2-pluq.C File Reference . . . . .	850
13.309.1 Function Documentation . . . . .	851
13.309.1.1 main() . . . . .	851
13.310 fflas-101_1.C File Reference . . . . .	851
13.310.1 Function Documentation . . . . .	851
13.310.1.1 main() . . . . .	851
13.311 fflas-101_3.C File Reference . . . . .	851
13.311.1 Function Documentation . . . . .	851
13.311.1.1 main() . . . . .	851
13.312 fflas_101.C File Reference . . . . .	852
13.312.1 Function Documentation . . . . .	852
13.312.1.1 main() . . . . .	852
13.313 fflas_101_lvl1.C File Reference . . . . .	852
13.313.1 Function Documentation . . . . .	852
13.313.1.1 main() . . . . .	852
13.314 ffpack-fgesv.C File Reference . . . . .	852
13.314.1 Function Documentation . . . . .	853
13.314.1.1 main() . . . . .	853
13.315 ffpack-solve.C File Reference . . . . .	853
13.315.1 Function Documentation . . . . .	853
13.315.1.1 main() . . . . .	853

# 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::fscale** (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::fscale** (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::fscalein** (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::fscalein** (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 assign 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\_redcoechelon** (**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 . . . . .	42
Interfaces . . . . .	42
Matrix Multiplication Algorithms . . . . .	42
SIMD wrapper . . . . .	42
FFPACK . . . . .	43
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	206
FFLAS::FieldCategories	
Traits and categories will need to be placed in a proper file later	206
FFLAS::MMHelperAlgo	207
FFLAS::ModeCategories	
Specifies the mode of action for an algorithm w.r.t	207
FFLAS::ParSeqHelper	
ParSeqHelper for both fgemm and ftrsm	208
FFLAS::Protected	208
FFLAS::sell_details	222
FFLAS::sparse_details	223
FFLAS::sparse_details_impl	237
FFLAS::StrategyParameter	277
FFLAS::StructureHelper	
StructureHelper for ftrsm	277
FFLAS::vectorised	278
FFLAS::vectorised::unswitch	283
FFPACK	
Finite Field <b>PACK</b> Set of elimination based routines for dense linear algebra	286
FFPACK::Protected	389
Givaro	396
MKL_CONFIG	396
RecInt	396



## Chapter 7

# Hierarchical Index

### 7.1 Class Hierarchy

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

AlgoChooser< ModeT, ParSeq > . . . . .	397
AlgoChooser< ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq > . . . . .	397
ALL< v > . . . . .	397
ALL< false, v... > . . . . .	397
ALL< true, v... > . . . . .	397
ALL<> . . . . .	397
ArbitraryPrecIntTag . . . . .	397
AreEqual< X, Y > . . . . .	397
AreEqual< X, X > . . . . .	397
Argument . . . . .	398
array< T > . . . . .	398
associatedDelayedField< Field > . . . . .	399
associatedDelayedField< const FFPACK::RNSIntegerMod< RNS > > . . . . .	399
associatedDelayedField< const Givaro::Modular< T, X > > . . . . .	399
associatedDelayedField< const Givaro::ModularBalanced< T > > . . . . .	399
associatedDelayedField< const Givaro::ZRing< T > > . . . . .	399
Auto . . . . .	399
Bench< Elt > . . . . .	399
Bini . . . . .	402
Block . . . . .	402
BlockTransposeSIMD< Field, Simd, > . . . . .	402
callLUdivine_small< Element > . . . . .	404
callLUdivine_small< double > . . . . .	404
callLUdivine_small< float > . . . . .	404
CharpolyFailed . . . . .	404
Checker_Empty< Field > . . . . .	405
CheckerImplem_charpoly< Field, Polynomial > . . . . .	405
CheckerImplem_charpoly< Givaro::ZRing< Givaro::Integer >, Polynomial > . . . . .	405
CheckerImplem_Det< Field > . . . . .	406
CheckerImplem_fgemm< Field > . . . . .	407
CheckerImplem_ftdsm< Field > . . . . .	408
CheckerImplem_invert< Field > . . . . .	409
CheckerImplem_PLUQ< Field > . . . . .	410
Classic . . . . .	411
Column . . . . .	411

CompactElement< Element > . . . . .	411
CompactElement< double > . . . . .	411
CompactElement< float > . . . . .	411
CompactElement< int16_t > . . . . .	411
CompactElement< int32_t > . . . . .	411
CompactElement< int64_t > . . . . .	411
compatible_data_type< Field > . . . . .	412
compatible_data_type< Givaro::ZRing< double > > . . . . .	412
compatible_data_type< Givaro::ZRing< float > > . . . . .	412
Compose< H1, H2 > . . . . .	412
array< T >::const_iterator . . . . .	413
string::const_iterator . . . . .	413
vector< T >::const_iterator . . . . .	413
array< T >::const_reverse_iterator . . . . .	413
string::const_reverse_iterator . . . . .	413
vector< T >::const_reverse_iterator . . . . .	414
Simd128_impl< true, true, false, 2 >::Converter . . . . .	497
Simd128_impl< true, true, false, 4 >::Converter . . . . .	497
Simd128_impl< true, true, false, 8 >::Converter . . . . .	497
Simd128_impl< true, true, true, 2 >::Converter . . . . .	497
Simd128_impl< true, true, true, 4 >::Converter . . . . .	497
Simd128_impl< true, true, true, 8 >::Converter . . . . .	497
Simd256_impl< true, false, true, 8 >::Converter . . . . .	498
Simd256_impl< true, true, false, 2 >::Converter . . . . .	498
Simd256_impl< true, true, false, 4 >::Converter . . . . .	498
Simd256_impl< true, true, false, 8 >::Converter . . . . .	498
Simd256_impl< true, true, true, 2 >::Converter . . . . .	498
Simd256_impl< true, true, true, 4 >::Converter . . . . .	498
Simd256_impl< true, true, true, 8 >::Converter . . . . .	498
Simd512_impl< true, true, false, 8 >::Converter . . . . .	499
Simd512_impl< true, true, true, 8 >::Converter . . . . .	499
ConvertTo< T > . . . . .	414
ConvertTo< ElementCategories::MachineFloatTag > . . . . .	414
ConvertTo< ElementCategories::RNSElementTag > . . . . .	414
Coo< ValT, IdxT > . . . . .	414
Coo< Field > . . . . .	416
Coo< ValT, IdxT > . . . . .	417
CooMat< Field > . . . . .	418
CooMat< FFPACK::RNSInteger > . . . . .	418
count_nonconst_lvalue_reference< T > . . . . .	419
count_nonconst_lvalue_reference< const T &, O... > . . . . .	419
count_nonconst_lvalue_reference< T &, O... > . . . . .	419
count_nonconst_lvalue_reference< T, O... > . . . . .	419
count_nonconst_lvalue_reference<> . . . . .	419
CsrMat< Field > . . . . .	419
CsrMat< FFPACK::RNSInteger > . . . . .	419
DefaultBoundedTag . . . . .	420
DefaultTag . . . . .	420
DelayedTag . . . . .	421
DivideAndConquer . . . . .	421
ElementTraits< Element > . . . . .	421
ElementTraits< double > . . . . .	421
ElementTraits< FFPACK::rns_double_elt > . . . . .	421
ElementTraits< float > . . . . .	421
ElementTraits< Givaro::Integer > . . . . .	421
ElementTraits< int16_t > . . . . .	421
ElementTraits< int32_t > . . . . .	421
ElementTraits< int64_t > . . . . .	421

ElementTraits< int8_t > . . . . .	421
ElementTraits< Reclnt::rint< K > > . . . . .	421
ElementTraits< Reclnt::rmint< K, MG > > . . . . .	421
ElementTraits< Reclnt::ruint< K > > . . . . .	421
ElementTraits< uint16_t > . . . . .	421
ElementTraits< uint32_t > . . . . .	421
ElementTraits< uint64_t > . . . . .	421
ElementTraits< uint8_t > . . . . .	421
ElIMat< Field > . . . . .	421
ElIMat< FFPACK::RNSInteger > . . . . .	421
Failure . . . . .	422
FailureCharpolyCheck . . . . .	424
FailureDetCheck . . . . .	424
FailureFgemmCheck . . . . .	424
FailureInvertCheck . . . . .	424
FailurePLUQCheck . . . . .	424
FailureTrsmCheck . . . . .	424
false_type	
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO > > . . . . .	449
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO_ZO > > . . . . .	449
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR > > . . . . .	449
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_HYB > > . . . . .	449
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_ZO > > . . . . .	449
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL > > . . . . .	449
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_ZO > > . . . . .	449
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd > > . . . . .	449
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > > . . . . .	449
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::HYB_ZO > > . . . . .	449
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL > > . . . . .	449
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL_ZO > > . . . . .	449
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO_ZO > > . . . . .	450
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_ZO > > . . . . .	450
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_ZO > > . . . . .	450
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > > . . . . .	450
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL_ZO > > . . . . .	450
support_fast_mod< double > . . . . .	504
support_fast_mod< float > . . . . .	504
support_fast_mod< int64_t > . . . . .	504
isSparseMatrix< Field, M > . . . . .	449
isSparseMatrixMKLFormat< F, M > . . . . .	449
isSparseMatrixSimdFormat< F, M > . . . . .	449
isZOSparseMatrix< F, M > . . . . .	450
support_fast_mod< T > . . . . .	504
support_simd< T > . . . . .	504
support_simd_add< T > . . . . .	505
support_simd_mod< T > . . . . .	505
field< constField > . . . . .	399
FieldSimd< _Field > . . . . .	425
FieldSimd< Field > . . . . .	425
FieldTraits< Field > . . . . .	431
FieldTraits< FFPACK::RNSInteger< T > > . . . . .	431
FieldTraits< FFPACK::RNSIntegerMod< T > > . . . . .	431
FieldTraits< Givaro::Modular< Element > > . . . . .	431
FieldTraits< Givaro::ModularBalanced< Element > > . . . . .	431
FieldTraits< Givaro::ZRing< double > > . . . . .	431
FieldTraits< Givaro::ZRing< float > > . . . . .	431
FieldTraits< Givaro::ZRing< Givaro::Integer > > . . . . .	431
FieldTraits< Givaro::ZRing< int16_t > > . . . . .	431

FieldTraits< Givaro::ZRing< int32_t > > . . . . .	431
FieldTraits< Givaro::ZRing< int64_t > > . . . . .	431
FieldTraits< Givaro::ZRing< Reclnt::ruint< K > > > . . . . .	431
FieldTraits< Givaro::ZRing< uint16_t > > . . . . .	431
FieldTraits< Givaro::ZRing< uint32_t > > . . . . .	431
FieldTraits< Givaro::ZRing< uint64_t > > . . . . .	431
Fixed . . . . .	431
FixedPrecIntTag . . . . .	431
ScalFunctionsBase< Element, typename enable_if< is_floating_point< Element >::value >::type >↔ ::FloatingPointTestDistribution . . . . .	495
ForStrategy1D< blocksize_t, Cut, Param > . . . . .	432
ForStrategy2D< blocksize_t, Cut, Param > . . . . .	434
ftmmLeftLowerNoTransNonUnit< Element > . . . . .	438
ftmmLeftLowerNoTransUnit< Element > . . . . .	438
ftmmLeftLowerTransNonUnit< Element > . . . . .	438
ftmmLeftLowerTransUnit< Element > . . . . .	438
ftmmLeftUpperNoTransNonUnit< Element > . . . . .	438
ftmmLeftUpperNoTransUnit< Element > . . . . .	439
ftmmLeftUpperTransNonUnit< Element > . . . . .	439
ftmmLeftUpperTransUnit< Element > . . . . .	439
ftmmRightLowerNoTransNonUnit< Element > . . . . .	439
ftmmRightLowerNoTransUnit< Element > . . . . .	439
ftmmRightLowerTransNonUnit< Element > . . . . .	439
ftmmRightLowerTransUnit< Element > . . . . .	439
ftmmRightUpperNoTransNonUnit< Element > . . . . .	439
ftmmRightUpperNoTransUnit< Element > . . . . .	440
ftmmRightUpperTransNonUnit< Element > . . . . .	440
ftmmRightUpperTransUnit< Element > . . . . .	440
ftsmLeftLowerNoTransNonUnit< Element > . . . . .	440
ftsmLeftLowerNoTransUnit< Element > . . . . .	440
ftsmLeftLowerTransNonUnit< Element > . . . . .	440
ftsmLeftLowerTransUnit< Element > . . . . .	440
ftsmLeftUpperNoTransNonUnit< Element > . . . . .	440
ftsmLeftUpperNoTransUnit< Element > . . . . .	441
ftsmLeftUpperTransNonUnit< Element > . . . . .	441
ftsmLeftUpperTransUnit< Element > . . . . .	441
ftsmRightLowerNoTransNonUnit< Element > . . . . .	441
ftsmRightLowerNoTransUnit< Element > . . . . .	441
ftsmRightLowerTransNonUnit< Element > . . . . .	442
ftsmRightLowerTransUnit< Element > . . . . .	442
ftsmRightUpperNoTransNonUnit< Element > . . . . .	442
ftsmRightUpperNoTransUnit< Element > . . . . .	442
ftsmRightUpperTransNonUnit< Element > . . . . .	442
ftsmRightUpperTransUnit< Element > . . . . .	442
GenericTag . . . . .	442
GenericTag . . . . .	443
Grain . . . . .	443
has_minus_eq_impl< C > . . . . .	443
has_minus_impl< C > . . . . .	443
has_mul_eq_impl< C > . . . . .	443
has_mul_impl< C > . . . . .	444
has_operation< T > . . . . .	444
has_plus_eq_impl< C > . . . . .	444
has_plus_impl< C > . . . . .	445
HelperFlag . . . . .	445
HelperMod< Field, ElementTraits > . . . . .	446
HelperMod< Field, ElementCategories::MachineIntTag > . . . . .	446
HelperMod< Field, FFLAS::ElementCategories::ArbitraryPrecIntTag > . . . . .	446



HelperMod< Field, FFLAS::ElementCategories::FixedPrecIntTag > . . . . .	446
HelperMod< Field, FFLAS::ElementCategories::MachineFloatTag > . . . . .	446
Hybrid . . . . .	446
Info . . . . .	446
Info . . . . .	447
is_all_same< Args > . . . . .	448
is_all_same< T, Args... > . . . . .	448
is_all_same<> . . . . .	448
is_simd< T > . . . . .	448
Iterative . . . . .	450
array< T >::iterator . . . . .	450
string::iterator . . . . .	450
vector< T >::iterator . . . . .	450
LazyTag . . . . .	450
limits< T > . . . . .	450
limits< char > . . . . .	450
limits< double > . . . . .	450
limits< float > . . . . .	450
limits< Givaro::Integer > . . . . .	450
limits< int > . . . . .	450
limits< long > . . . . .	450
limits< long long > . . . . .	450
limits< ReclInt::rint< K > > . . . . .	450
limits< ReclInt::ruint< K > > . . . . .	450
limits< short int > . . . . .	450
limits< signed char > . . . . .	450
limits< unsigned char > . . . . .	450
limits< unsigned int > . . . . .	450
limits< unsigned long > . . . . .	450
limits< unsigned long long > . . . . .	450
limits< unsigned short int > . . . . .	450
MachineFloatTag . . . . .	451
MachineIntTag . . . . .	451
MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait > . . . . .	451
MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait > . . . . .	451
MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait > . . . . .	451
MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< Dest >, ParSeqTrait > . . . . .	451
MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeqTrait > . . . . .	451
MMHelper< Field, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait > . . . . .	451
ModeTraits< Field > . . . . .	456
ModeTraits< Givaro::Modular< Element, Compute > > . . . . .	456
ModeTraits< Givaro::Modular< Givaro::Integer, Compute > > . . . . .	456
ModeTraits< Givaro::Modular< int16_t, Compute > > . . . . .	456
ModeTraits< Givaro::Modular< int32_t, Compute > > . . . . .	456
ModeTraits< Givaro::Modular< int64_t, uint64_t > > . . . . .	456
ModeTraits< Givaro::Modular< int8_t, Compute > > . . . . .	456
ModeTraits< Givaro::Modular< ReclInt::ruint< K >, Compute > > . . . . .	456
ModeTraits< Givaro::Modular< uint16_t, Compute > > . . . . .	456
ModeTraits< Givaro::Modular< uint32_t, Compute > > . . . . .	456
ModeTraits< Givaro::Modular< uint8_t, Compute > > . . . . .	456
ModeTraits< Givaro::ModularBalanced< Element > > . . . . .	456
ModeTraits< Givaro::ModularBalanced< Givaro::Integer > > . . . . .	456
ModeTraits< Givaro::ModularBalanced< int16_t > > . . . . .	456
ModeTraits< Givaro::ModularBalanced< int32_t > > . . . . .	456
ModeTraits< Givaro::ModularBalanced< int8_t > > . . . . .	456
ModeTraits< Givaro::Montgomery< T > > . . . . .	456
ModeTraits< Givaro::ZRing< double > > . . . . .	456

ModeTraits< Givaro::ZRing< float > > . . . . .	456
ModeTraits< Givaro::ZRing< Givaro::Integer > > . . . . .	456
ModularBalanced< T > . . . . .	457
ModularBalanced< T > . . . . .	457
ModularBalanced< double > . . . . .	457
ModularTag . . . . .	457
Montgomery< T > . . . . .	457
need_field_characteristic< Field > . . . . .	457
need_field_characteristic< Givaro::Modular< Field > > . . . . .	457
need_field_characteristic< Givaro::ModularBalanced< Field > > . . . . .	457
NoSimd< T > . . . . .	457
Parallel< C, P > . . . . .	459
readMyMachineType< Field, T > . . . . .	462
readMyMachineType< Field, mpz_t > . . . . .	462
Recursive . . . . .	463
Recursive . . . . .	463
array< T >::reverse_iterator . . . . .	463
string::reverse_iterator . . . . .	463
vector< T >::reverse_iterator . . . . .	463
rint< K > . . . . .	463
rns_double . . . . .	463
rns_double_elt . . . . .	468
rns_double_elt_cstptr . . . . .	470
rns_double_elt_ptr . . . . .	472
rns_double_extended . . . . .	475
RNSElementTag . . . . .	479
RNSInteger< RNS > . . . . .	479
RNSInteger< FFPACK::rns_double > . . . . .	479
RNSIntegerMod< RNS > . . . . .	482
RNSIntegerMod< FFPACK::rns_double > . . . . .	482
rnsRandIter< RNS > . . . . .	489
RNSInteger< RNS >::RandIter . . . . .	460
RNSIntegerMod< RNS >::RandIter . . . . .	461
Row . . . . .	490
ruint< K > . . . . .	490
ScalFunctionsBase< Element, Enable > . . . . .	495
ScalFunctions< Element > . . . . .	490
ScalFunctionsBase< Element, typename enable_if< is_floating_point< Element >::value >::type > . . . . .	495
ScalFunctionsBase< Element, typename enable_if< is_integral< Element >::value >::type > . . . . .	495
Sequential . . . . .	496
Simd128_impl< ArithType, Int, Signed, Size > . . . . .	497
Simd128_impl< std::is_arithmetic< T >::value, std::is_integral< T >::value, std::is_signed< T >::value, sizeof(T)> . . . . .	497
Simd128_impl< true, false, true, 4 > . . . . .	497
Simd128_impl< true, false, true, 8 > . . . . .	497
Simd128i_base . . . . .	497
Simd128_impl< true, true, true, 2 > . . . . .	497
Simd128_impl< true, true, false, 2 > . . . . .	497
Simd128_impl< true, true, true, 4 > . . . . .	497
Simd128_impl< true, true, false, 4 > . . . . .	497
Simd128_impl< true, true, true, 8 > . . . . .	497
Simd128_impl< true, true, false, 8 > . . . . .	497
Simd256_impl< ArithType, Int, Signed, Size > . . . . .	498
Simd256_impl< std::is_arithmetic< T >::value, std::is_integral< T >::value, std::is_signed< T >::value, sizeof(T)> . . . . .	498
Simd256fp_base . . . . .	498
Simd256_impl< true, false, true, 4 > . . . . .	498

Simd256_impl< true, false, true, 8 > . . . . .	498
Simd256i_base . . . . .	498
Simd256_impl< true, true, true, 2 > . . . . .	498
Simd256_impl< true, true, false, 2 > . . . . .	498
Simd256_impl< true, true, true, 4 > . . . . .	498
Simd256_impl< true, true, false, 4 > . . . . .	498
Simd256_impl< true, true, false, 4 > . . . . .	498
Simd256_impl< true, true, true, 8 > . . . . .	498
Simd256_impl< true, true, false, 8 > . . . . .	498
Simd512_impl< ArithType, Int, Signed, Size > . . . . .	499
Simd512_impl< std::is_arithmetic< T >::value, std::is_integral< T >::value, std::is_signed< T >::value, sizeof(T)> . . . . .	499
Simd512_impl< true, false, true, 4 > . . . . .	499
Simd512_impl< true, false, true, 8 > . . . . .	499
Simd512i_base . . . . .	499
Simd256_impl< true, true, true, 4 > . . . . .	498
Simd512_impl< true, true, true, 8 > . . . . .	499
Simd512_impl< true, true, false, 8 > . . . . .	499
SimdChooser< T, bool, bool > . . . . .	500
SimdChooser< T, false, b > . . . . .	500
SimdChooser< T, true, false > . . . . .	500
SimdChooser< T, true, true > . . . . .	500
simdToType< T > . . . . .	500
Single . . . . .	500
Sparse< Field, SparseMatrix_t, IdxT, PtrT > . . . . .	500
Sparse< _Field, SparseMatrix_t::COO > . . . . .	500
Sparse< _Field, SparseMatrix_t::COO_ZO > . . . . .	500
Sparse< _Field, SparseMatrix_t::CSR > . . . . .	500
Sparse< _Field, SparseMatrix_t::CSR_ZO > . . . . .	500
Sparse< _Field, SparseMatrix_t::CSR_HYB > . . . . .	500
Sparse< _Field, SparseMatrix_t::ELL > . . . . .	500
Sparse< _Field, SparseMatrix_t::ELL_ZO > . . . . .	500
Sparse< _Field, SparseMatrix_t::ELL_simd > . . . . .	500
Sparse< _Field, SparseMatrix_t::ELL_simd_ZO > . . . . .	500
Sparse< _Field, SparseMatrix_t::HYB_ZO > . . . . .	500
Sparse< _Field, SparseMatrix_t::SELL > . . . . .	500
Sparse< _Field, SparseMatrix_t::SELL_ZO > . . . . .	500
Sparse< FFPACK::RNSInteger, SparseMatrix_t::COO, int16_t > . . . . .	500
Sparse< FFPACK::RNSInteger, SparseMatrix_t::COO, int32_t > . . . . .	500
Sparse< FFPACK::RNSInteger, SparseMatrix_t::COO, int64_t > . . . . .	500
Sparse< FFPACK::RNSInteger, SparseMatrix_t::COO_ZO, int16_t > . . . . .	500
Sparse< FFPACK::RNSInteger, SparseMatrix_t::COO_ZO, int32_t > . . . . .	500
Sparse< FFPACK::RNSInteger, SparseMatrix_t::COO_ZO, int64_t > . . . . .	500
Sparse< FFPACK::RNSInteger, SparseMatrix_t::CSR, int16_t > . . . . .	500
Sparse< FFPACK::RNSInteger, SparseMatrix_t::CSR, int32_t > . . . . .	500
Sparse< FFPACK::RNSInteger, SparseMatrix_t::CSR, int64_t > . . . . .	500
Sparse< FFPACK::RNSInteger, SparseMatrix_t::CSR_ZO, int16_t > . . . . .	500
Sparse< FFPACK::RNSInteger, SparseMatrix_t::CSR_ZO, int32_t > . . . . .	500
Sparse< FFPACK::RNSInteger, SparseMatrix_t::CSR_ZO, int64_t > . . . . .	500
Sparse< FFPACK::RNSInteger, SparseMatrix_t::ELL, int16_t > . . . . .	500
Sparse< FFPACK::RNSInteger, SparseMatrix_t::ELL, int32_t > . . . . .	500
Sparse< FFPACK::RNSInteger, SparseMatrix_t::ELL, int64_t > . . . . .	500
Sparse< FFPACK::RNSInteger, SparseMatrix_t::ELL_ZO, int16_t > . . . . .	500
Sparse< FFPACK::RNSInteger, SparseMatrix_t::ELL_ZO, int32_t > . . . . .	500
Sparse< FFPACK::RNSInteger, SparseMatrix_t::ELL_ZO, int64_t > . . . . .	500
SpMat< Field, flag > . . . . .	501

StatsMatrix	501
string	504
Test< Elt >	505
TestOneMethod< Simd >	508
tfn_minus	511
tfn_minus_eq	511
tfn_mul	512
tfn_mul_eq	512
tfn_plus	512
tfn_plus_eq	513
Threads	513
ThreeD	513
ThreeDAdaptive	513
ThreeDInPlace	514
TRSMHelper< ReclterTrait, ParSeqTrait >	514
true_type	
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO > >	449
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO_ZO > >	449
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR > >	449
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_HYB > >	449
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_ZO > >	449
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL > >	449
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_ZO > >	449
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd > >	449
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > >	449
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::HYB_ZO > >	449
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL > >	449
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL_ZO > >	449
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO_ZO > >	450
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_ZO > >	450
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_ZO > >	450
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > >	450
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL_ZO > >	450
support_fast_mod< double >	504
support_fast_mod< float >	504
support_fast_mod< int64_t >	504
TwoD	515
TwoDAdaptive	515
type< constField >	399
UnparametricTag	515
vector< T >	516
width< T >	516
width< double >	516
width< float >	516
Winograd	516
WinogradPar	516

## 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>	397
<a href="#">AlgoChooser&lt; ModeCategories::ConvertTo&lt; ElementCategories::RNSElementTag &gt;, ParSeq &gt;</a>	397
<a href="#">ALL&lt; v &gt;</a>	397
<a href="#">ALL&lt; false, v... &gt;</a>	397
<a href="#">ALL&lt; true, v... &gt;</a>	397
<a href="#">ALL&lt;&gt;</a>	397
<a href="#">ArbitraryPrecIntTag</a>	
Arbitrary precision integers: GMP	397
<a href="#">AreEqual&lt; X, Y &gt;</a>	397
<a href="#">AreEqual&lt; X, X &gt;</a>	397
<a href="#">Argument</a>	398
<a href="#">array&lt; T &gt;</a>	
STL class	398
<a href="#">associatedDelayedField&lt; Field &gt;</a>	399
<a href="#">associatedDelayedField&lt; const FFPACK::RNSIntegerMod&lt; RNS &gt; &gt;</a>	399
<a href="#">associatedDelayedField&lt; const Givaro::Modular&lt; T, X &gt; &gt;</a>	399
<a href="#">associatedDelayedField&lt; const Givaro::ModularBalanced&lt; T &gt; &gt;</a>	399
<a href="#">associatedDelayedField&lt; const Givaro::ZRing&lt; T &gt; &gt;</a>	399
<a href="#">Auto</a>	399
<a href="#">Bench&lt; Elt &gt;</a>	399
<a href="#">Bini</a>	402
<a href="#">Block</a>	402
<a href="#">BlockTransposeSIMD&lt; Field, Simd, &gt;</a>	402
<a href="#">callLUdivine_small&lt; Element &gt;</a>	404
<a href="#">callLUdivine_small&lt; double &gt;</a>	404
<a href="#">callLUdivine_small&lt; float &gt;</a>	404
<a href="#">CharpolyFailed</a>	404
<a href="#">Checker_Empty&lt; Field &gt;</a>	405
<a href="#">CheckerImplem_charpoly&lt; Field, Polynomial &gt;</a>	405
<a href="#">CheckerImplem_charpoly&lt; Givaro::ZRing&lt; Givaro::Integer &gt;, Polynomial &gt;</a>	405
<a href="#">CheckerImplem_Det&lt; Field &gt;</a>	406
<a href="#">CheckerImplem_fgemm&lt; Field &gt;</a>	407
<a href="#">CheckerImplem_ftdsm&lt; Field &gt;</a>	408
<a href="#">CheckerImplem_invert&lt; Field &gt;</a>	409
<a href="#">CheckerImplem_PLUQ&lt; Field &gt;</a>	410

Classic	411
Column	411
CompactElement< Element >	411
CompactElement< double >	411
CompactElement< float >	411
CompactElement< int16_t >	411
CompactElement< int32_t >	411
CompactElement< int64_t >	411
compatible_data_type< Field >	412
compatible_data_type< Givaro::ZRing< double > >	412
compatible_data_type< Givaro::ZRing< float > >	412
Compose< H1, H2 >	412
array< T >::const_iterator	413
string::const_iterator	413
vector< T >::const_iterator	413
array< T >::const_reverse_iterator	413
string::const_reverse_iterator	413
vector< T >::const_reverse_iterator	414
Simd128_impl< true, true, false, 2 >::Converter	497
Simd128_impl< true, true, false, 4 >::Converter	497
Simd128_impl< true, true, false, 8 >::Converter	497
Simd128_impl< true, true, true, 2 >::Converter	497
Simd128_impl< true, true, true, 4 >::Converter	497
Simd128_impl< true, true, true, 8 >::Converter	497
Simd256_impl< true, false, true, 8 >::Converter	498
Simd256_impl< true, true, false, 2 >::Converter	498
Simd256_impl< true, true, false, 4 >::Converter	498
Simd256_impl< true, true, false, 8 >::Converter	498
Simd256_impl< true, true, true, 2 >::Converter	498
Simd256_impl< true, true, true, 4 >::Converter	498
Simd256_impl< true, true, true, 8 >::Converter	498
Simd512_impl< true, true, false, 8 >::Converter	499
Simd512_impl< true, true, true, 8 >::Converter	499
ConvertTo< T >	
Force conversion to appropriate element type of ElementCategory T	414
Coo< ValT, IdxT >	414
Coo< Field >	416
Coo< ValT, IdxT >	417
CooMat< Field >	418
count_nonconst_lvalue_reference< T >	419
count_nonconst_lvalue_reference< const T &, O... >	419
count_nonconst_lvalue_reference< T &, O... >	419
count_nonconst_lvalue_reference< T, O... >	419
count_nonconst_lvalue_reference<>	419
CsrMat< Field >	419
DefaultBoundedTag	
Use standard field operations, but keeps track of bounds on input and output	420
DefaultTag	
No specific mode of action: use standard field operations	420
DelayedTag	
Performs field operations with delayed mod reductions. Ensures result is reduced	421
DivideAndConquer	421
ElementTraits< Element >	
ElementTraits	421
ElementTraits< double >	421
ElementTraits< FFPACK::rns_double_elt >	421
ElementTraits< float >	421
ElementTraits< Givaro::Integer >	421

ElementTraits< int16_t > . . . . .	421
ElementTraits< int32_t > . . . . .	421
ElementTraits< int64_t > . . . . .	421
ElementTraits< int8_t > . . . . .	421
ElementTraits< Reclnt::rint< K > > . . . . .	421
ElementTraits< Reclnt::rmint< K, MG > > . . . . .	421
ElementTraits< Reclnt::ruint< K > > . . . . .	421
ElementTraits< uint16_t > . . . . .	421
ElementTraits< uint32_t > . . . . .	421
ElementTraits< uint64_t > . . . . .	421
ElementTraits< uint8_t > . . . . .	421
EllMat< Field > . . . . .	421
Failure	
A precondition failed . . . . .	422
FailureCharpolyCheck . . . . .	424
FailureDetCheck . . . . .	424
FailureFgemmCheck . . . . .	424
FailureInvertCheck . . . . .	424
FailurePLUQCheck . . . . .	424
FailureTrsmCheck . . . . .	424
FieldSimd< _Field > . . . . .	425
FieldTraits< Field >	
FieldTrait . . . . .	431
FieldTraits< FFPACK::RNSInteger< T > > . . . . .	431
FieldTraits< FFPACK::RNSIntegerMod< T > > . . . . .	431
FieldTraits< Givaro::Modular< Element > > . . . . .	431
FieldTraits< Givaro::ModularBalanced< Element > > . . . . .	431
FieldTraits< Givaro::ZRing< double > > . . . . .	431
FieldTraits< Givaro::ZRing< float > > . . . . .	431
FieldTraits< Givaro::ZRing< Givaro::Integer > > . . . . .	431
FieldTraits< Givaro::ZRing< int16_t > > . . . . .	431
FieldTraits< Givaro::ZRing< int32_t > > . . . . .	431
FieldTraits< Givaro::ZRing< int64_t > > . . . . .	431
FieldTraits< Givaro::ZRing< Reclnt::ruint< K > > > . . . . .	431
FieldTraits< Givaro::ZRing< uint16_t > > . . . . .	431
FieldTraits< Givaro::ZRing< uint32_t > > . . . . .	431
FieldTraits< Givaro::ZRing< uint64_t > > . . . . .	431
Fixed . . . . .	431
FixedPrecIntTag	
Fixed precision integers above machine precision: Givaro::reclnt . . . . .	431
ScalFunctionsBase< Element, typename enable_if< is_floating_point< Element >::value >::type >::FloatingPointTestDistribution	
495	
ForStrategy1D< blocksize_t, Cut, Param > . . . . .	432
ForStrategy2D< blocksize_t, Cut, Param > . . . . .	434
ftmmLeftLowerNoTransNonUnit< Element > . . . . .	438
ftmmLeftLowerNoTransUnit< Element > . . . . .	438
ftmmLeftLowerTransNonUnit< Element > . . . . .	438
ftmmLeftLowerTransUnit< Element > . . . . .	438
ftmmLeftUpperNoTransNonUnit< Element > . . . . .	438
ftmmLeftUpperNoTransUnit< Element > . . . . .	439
ftmmLeftUpperTransNonUnit< Element > . . . . .	439
ftmmLeftUpperTransUnit< Element > . . . . .	439
ftmmRightLowerNoTransNonUnit< Element > . . . . .	439
ftmmRightLowerNoTransUnit< Element > . . . . .	439
ftmmRightLowerTransNonUnit< Element > . . . . .	439
ftmmRightLowerTransUnit< Element > . . . . .	439
ftmmRightUpperNoTransNonUnit< Element > . . . . .	439
ftmmRightUpperNoTransUnit< Element > . . . . .	440

<a href="#">ftrmmRightUpperTransNonUnit&lt; Element &gt;</a>	440
<a href="#">ftrmmRightUpperTransUnit&lt; Element &gt;</a>	440
<a href="#">ftrsmLeftLowerNoTransNonUnit&lt; Element &gt;</a>	440
<a href="#">ftrsmLeftLowerNoTransUnit&lt; Element &gt;</a>	440
<a href="#">ftrsmLeftLowerTransNonUnit&lt; Element &gt;</a>	440
<a href="#">ftrsmLeftLowerTransUnit&lt; Element &gt;</a>	440
<a href="#">ftrsmLeftUpperNoTransNonUnit&lt; Element &gt;</a>	440
Computes the maximal size for delaying the modular reduction in a triangular system resolution	440
<a href="#">ftrsmLeftUpperNoTransUnit&lt; Element &gt;</a>	441
<a href="#">ftrsmLeftUpperTransNonUnit&lt; Element &gt;</a>	441
<a href="#">ftrsmLeftUpperTransUnit&lt; Element &gt;</a>	441
<a href="#">ftrsmRightLowerNoTransNonUnit&lt; Element &gt;</a>	441
<a href="#">ftrsmRightLowerNoTransUnit&lt; Element &gt;</a>	441
<a href="#">ftrsmRightLowerTransNonUnit&lt; Element &gt;</a>	442
<a href="#">ftrsmRightLowerTransUnit&lt; Element &gt;</a>	442
<a href="#">ftrsmRightUpperNoTransNonUnit&lt; Element &gt;</a>	442
<a href="#">ftrsmRightUpperNoTransUnit&lt; Element &gt;</a>	442
<a href="#">ftrsmRightUpperTransNonUnit&lt; Element &gt;</a>	442
<a href="#">ftrsmRightUpperTransUnit&lt; Element &gt;</a>	442
<a href="#">GenericTag</a>	
Default is generic	442
<a href="#">GenericTag</a>	
Generic ring	443
<a href="#">Grain</a>	443
<a href="#">has_minus_eq_impl&lt; C &gt;</a>	443
<a href="#">has_minus_impl&lt; C &gt;</a>	443
<a href="#">has_mul_eq_impl&lt; C &gt;</a>	443
<a href="#">has_mul_impl&lt; C &gt;</a>	444
<a href="#">has_operation&lt; T &gt;</a>	444
<a href="#">has_plus_eq_impl&lt; C &gt;</a>	444
<a href="#">has_plus_impl&lt; C &gt;</a>	445
<a href="#">HelperFlag</a>	445
<a href="#">HelperMod&lt; Field, ElementTraits &gt;</a>	446
<a href="#">HelperMod&lt; Field, ElementCategories::MachineIntTag &gt;</a>	446
<a href="#">HelperMod&lt; Field, FFLAS::ElementCategories::ArbitraryPrecIntTag &gt;</a>	446
<a href="#">HelperMod&lt; Field, FFLAS::ElementCategories::FixedPrecIntTag &gt;</a>	446
<a href="#">HelperMod&lt; Field, FFLAS::ElementCategories::MachineFloatTag &gt;</a>	446
<a href="#">Hybrid</a>	446
<a href="#">Info</a>	446
<a href="#">Info</a>	447
<a href="#">is_all_same&lt; Args &gt;</a>	448
<a href="#">is_all_same&lt; T, Args... &gt;</a>	448
<a href="#">is_all_same&lt;&gt;</a>	448
<a href="#">is_simd&lt; T &gt;</a>	448
<a href="#">isSparseMatrix&lt; Field, M &gt;</a>	449
<a href="#">isSparseMatrix&lt; Field, Sparse&lt; Field, SparseMatrix_t::COO &gt; &gt;</a>	449
<a href="#">isSparseMatrix&lt; Field, Sparse&lt; Field, SparseMatrix_t::COO_ZO &gt; &gt;</a>	449
<a href="#">isSparseMatrix&lt; Field, Sparse&lt; Field, SparseMatrix_t::CSR &gt; &gt;</a>	449
<a href="#">isSparseMatrix&lt; Field, Sparse&lt; Field, SparseMatrix_t::CSR_HYB &gt; &gt;</a>	449
<a href="#">isSparseMatrix&lt; Field, Sparse&lt; Field, SparseMatrix_t::CSR_ZO &gt; &gt;</a>	449
<a href="#">isSparseMatrix&lt; Field, Sparse&lt; Field, SparseMatrix_t::ELL &gt; &gt;</a>	449
<a href="#">isSparseMatrix&lt; Field, Sparse&lt; Field, SparseMatrix_t::ELL_simd &gt; &gt;</a>	449
<a href="#">isSparseMatrix&lt; Field, Sparse&lt; Field, SparseMatrix_t::ELL_simd_ZO &gt; &gt;</a>	449
<a href="#">isSparseMatrix&lt; Field, Sparse&lt; Field, SparseMatrix_t::ELL_ZO &gt; &gt;</a>	449
<a href="#">isSparseMatrix&lt; Field, Sparse&lt; Field, SparseMatrix_t::HYB_ZO &gt; &gt;</a>	449
<a href="#">isSparseMatrix&lt; Field, Sparse&lt; Field, SparseMatrix_t::SELL &gt; &gt;</a>	449
<a href="#">isSparseMatrix&lt; Field, Sparse&lt; Field, SparseMatrix_t::SELL_ZO &gt; &gt;</a>	449
<a href="#">isSparseMatrixMKLFormat&lt; F, M &gt;</a>	449



isSparseMatrixSimdFormat< F, M > . . . . .	449
isZOSparseMatrix< F, M > . . . . .	450
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO_ZO > > . . . . .	450
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_ZO > > . . . . .	450
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > > . . . . .	450
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_ZO > > . . . . .	450
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL_ZO > > . . . . .	450
Iterative . . . . .	450
array< T >::iterator . . . . .	450
string::iterator . . . . .	450
vector< T >::iterator . . . . .	450
LazyTag . . . . .	
Performs field operations with delayed mod only when necessary. Result may not be reduced . . . . .	450
limits< T > . . . . .	450
limits< char > . . . . .	450
limits< double > . . . . .	450
limits< float > . . . . .	450
limits< Givaro::Integer > . . . . .	450
limits< int > . . . . .	450
limits< long > . . . . .	450
limits< long long > . . . . .	450
limits< RecInt::rint< K > > . . . . .	450
limits< RecInt::ruint< K > > . . . . .	450
limits< short int > . . . . .	450
limits< signed char > . . . . .	450
limits< unsigned char > . . . . .	450
limits< unsigned int > . . . . .	450
limits< unsigned long > . . . . .	450
limits< unsigned long long > . . . . .	450
limits< unsigned short int > . . . . .	450
MachineFloatTag . . . . .	
Float or double . . . . .	451
MachineIntTag . . . . .	
Short, int, long, long long, and unsigned variants . . . . .	451
MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait > . . . . .	451
MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait > . . . . .	451
MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait > . . . . .	451
MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< Dest >, ParSeqTrait > . . . . .	451
MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeqTrait > . . . . .	451
451	
MMHelper< Field, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait > . . . . .	
FGEMM Helper for Default and ConvertTo modes of operation . . . . .	451
ModeTraits< Field > . . . . .	
ModeTraits . . . . .	456
ModeTraits< Givaro::Modular< Element, Compute > > . . . . .	456
ModeTraits< Givaro::Modular< Givaro::Integer, Compute > > . . . . .	456
ModeTraits< Givaro::Modular< int16_t, Compute > > . . . . .	456
ModeTraits< Givaro::Modular< int32_t, Compute > > . . . . .	456
ModeTraits< Givaro::Modular< int64_t, uint64_t > > . . . . .	456
ModeTraits< Givaro::Modular< int8_t, Compute > > . . . . .	456
ModeTraits< Givaro::Modular< RecInt::ruint< K >, Compute > > . . . . .	456
ModeTraits< Givaro::Modular< uint16_t, Compute > > . . . . .	456
ModeTraits< Givaro::Modular< uint32_t, Compute > > . . . . .	456
ModeTraits< Givaro::Modular< uint8_t, Compute > > . . . . .	456
ModeTraits< Givaro::ModularBalanced< Element > > . . . . .	456
ModeTraits< Givaro::ModularBalanced< Givaro::Integer > > . . . . .	456
ModeTraits< Givaro::ModularBalanced< int16_t > > . . . . .	456
ModeTraits< Givaro::ModularBalanced< int32_t > > . . . . .	456

ModeTraits< Givaro::ModularBalanced< int8_t > > . . . . .	456
ModeTraits< Givaro::Montgomery< T > > . . . . .	456
ModeTraits< Givaro::ZRing< double > > . . . . .	456
ModeTraits< Givaro::ZRing< float > > . . . . .	456
ModeTraits< Givaro::ZRing< Givaro::Integer > > . . . . .	456
ModularBalanced< T > . . . . .	457
ModularBalanced< T > . . . . .	457
ModularTag	
This is a modular field like e.g. Modular<T> or ModularBalanced<T> . . . . .	457
Montgomery< T > . . . . .	457
need_field_characteristic< Field > . . . . .	457
need_field_characteristic< Givaro::Modular< Field > > . . . . .	457
need_field_characteristic< Givaro::ModularBalanced< Field > > . . . . .	457
NoSimd< T > . . . . .	457
Parallel< C, P > . . . . .	459
RNSInteger< RNS >::RandIter . . . . .	460
RNSIntegerMod< RNS >::RandIter . . . . .	461
readMyMachineType< Field, T > . . . . .	462
readMyMachineType< Field, mpz_t > . . . . .	462
Recursive . . . . .	463
Recursive . . . . .	463
array< T >::reverse_iterator . . . . .	463
string::reverse_iterator . . . . .	463
vector< T >::reverse_iterator . . . . .	463
rint< K > . . . . .	463
rns_double . . . . .	463
rns_double_elt . . . . .	468
rns_double_elt_cstptr . . . . .	470
rns_double_elt_ptr . . . . .	472
rns_double_extended . . . . .	475
RNSElementTag	
Representation in a Residue Number System . . . . .	479
RNSInteger< RNS > . . . . .	479
RNSIntegerMod< RNS > . . . . .	482
rnsRandIter< RNS > . . . . .	489
Row . . . . .	490
ruint< K > . . . . .	490
ScalFunctions< Element > . . . . .	490
ScalFunctionsBase< Element, Enable > . . . . .	495
ScalFunctionsBase< Element, typename enable_if< is_floating_point< Element >::value >::type > . . . . .	495
ScalFunctionsBase< Element, typename enable_if< is_integral< Element >::value >::type > . . . . .	495
Sequential . . . . .	496
Simd128_impl< ArithType, Int, Signed, Size > . . . . .	497
Simd128_impl< true, false, true, 4 > . . . . .	497
Simd128_impl< true, false, true, 8 > . . . . .	497
Simd128_impl< true, true, false, 2 > . . . . .	497
Simd128_impl< true, true, false, 4 > . . . . .	497
Simd128_impl< true, true, false, 8 > . . . . .	497
Simd128_impl< true, true, true, 2 > . . . . .	497
Simd128_impl< true, true, true, 4 > . . . . .	497
Simd128_impl< true, true, true, 8 > . . . . .	497
Simd128i_base . . . . .	497
Simd256_impl< ArithType, Int, Signed, Size > . . . . .	498
Simd256_impl< true, false, true, 4 > . . . . .	498
Simd256_impl< true, false, true, 8 > . . . . .	498
Simd256_impl< true, true, false, 2 > . . . . .	498
Simd256_impl< true, true, false, 4 > . . . . .	498
Simd256_impl< true, true, false, 8 > . . . . .	498

Simd256_impl< true, true, true, 2 > . . . . .	498
Simd256_impl< true, true, true, 4 > . . . . .	498
Simd256_impl< true, true, true, 8 > . . . . .	498
Simd256fp_base . . . . .	498
Simd256i_base . . . . .	498
Simd512_impl< ArithType, Int, Signed, Size > . . . . .	499
Simd512_impl< true, false, true, 4 > . . . . .	499
Simd512_impl< true, false, true, 8 > . . . . .	499
Simd512_impl< true, true, false, 8 > . . . . .	499
Simd512_impl< true, true, true, 8 > . . . . .	499
Simd512i_base . . . . .	499
SimdChooser< T, bool, bool > . . . . .	500
SimdChooser< T, false, b > . . . . .	500
SimdChooser< T, true, false > . . . . .	500
SimdChooser< T, true, true > . . . . .	500
simdToType< T > . . . . .	500
Single . . . . .	500
Sparse< Field, SparseMatrix_t, IdxT, PtrT > . . . . .	500
Sparse< _Field, SparseMatrix_t::COO > . . . . .	500
Sparse< _Field, SparseMatrix_t::COO_ZO > . . . . .	500
Sparse< _Field, SparseMatrix_t::CSR > . . . . .	500
Sparse< _Field, SparseMatrix_t::CSR_HYB > . . . . .	500
Sparse< _Field, SparseMatrix_t::CSR_ZO > . . . . .	500
Sparse< _Field, SparseMatrix_t::ELL > . . . . .	500
Sparse< _Field, SparseMatrix_t::ELL_simd > . . . . .	500
Sparse< _Field, SparseMatrix_t::ELL_simd_ZO > . . . . .	500
Sparse< _Field, SparseMatrix_t::ELL_ZO > . . . . .	500
Sparse< _Field, SparseMatrix_t::HYB_ZO > . . . . .	500
Sparse< _Field, SparseMatrix_t::SELL > . . . . .	500
Sparse< _Field, SparseMatrix_t::SELL_ZO > . . . . .	500
SpMat< Field, flag > . . . . .	501
StatsMatrix . . . . .	501
string . . . . .	501
STL class . . . . .	504
support_fast_mod< T > . . . . .	504
support_fast_mod< double > . . . . .	504
support_fast_mod< float > . . . . .	504
support_fast_mod< int64_t > . . . . .	504
support_simd< T > . . . . .	504
support_simd_add< T > . . . . .	505
support_simd_mod< T > . . . . .	505
Test< Elt > . . . . .	505
TestOneMethod< Simd > . . . . .	508
tfn_minus . . . . .	511
tfn_minus_eq . . . . .	511
tfn_mul . . . . .	512
tfn_mul_eq . . . . .	512
tfn_plus . . . . .	512
tfn_plus_eq . . . . .	513
Threads . . . . .	513
ThreeD . . . . .	513
ThreeDAdaptive . . . . .	513
ThreeDInPlace . . . . .	514
TRSMHelper< RectIterTrait, ParSeqTrait > . . . . .	514
TRSM Helper . . . . .	514
TwoD . . . . .	515
TwoDAdaptive . . . . .	515

UnparametricTag	
If the field uses a representation with infix operators . . . . .	515
vector< T >	
STL class . . . . .	516
width< T > . . . . .	516
width< double > . . . . .	516
width< float > . . . . .	516
Winograd . . . . .	516
WinogradPar . . . . .	516

# Chapter 9

## File Index

### 9.1 File List

Here is a list of all files with brief descriptions:

<a href="#">arithprog.C</a>	517
<a href="#">autotune/charpoly.C</a>	517
<a href="#">examples/charpoly.C</a>	518
<a href="#">fsyrk.C</a>	519
<a href="#">fsytrf.C</a>	519
<a href="#">ftrtri.C</a>	520
<a href="#">autotune/pluq.C</a>	521
<a href="#">examples/pluq.C</a>	522
<a href="#">winograd.C</a>	522
<a href="#">benchmark-charpoly-mp.C</a>	523
<a href="#">benchmark-charpoly.C</a>	524
<a href="#">benchmark-checkers.C</a>	524
<a href="#">benchmark-dgemm.C</a>	525
<a href="#">benchmark-dgetrf.C</a>	526
<a href="#">benchmark-dgetri.C</a>	527
<a href="#">benchmark-dsytrf.C</a>	527
<a href="#">benchmark-dtrsm.C</a>	528
<a href="#">benchmark-dtrtri.C</a>	528
<a href="#">benchmark-fadd-lvl2.C</a>	529
<a href="#">benchmark-fdot.C</a>	529
<a href="#">benchmark-fgemm-mp.C</a>	530
<a href="#">benchmark-fgemm-rns.C</a>	531
<a href="#">benchmark-fgemm.C</a>	533
<a href="#">benchmark-fgemv-mp.C</a>	533
<a href="#">benchmark-fgemv.C</a>	535
<a href="#">benchmark-fgesv.C</a>	538
<a href="#">benchmark-fsyr2k.C</a>	538
<a href="#">benchmark-fsyrk.C</a>	539
<a href="#">benchmark-fsytrf.C</a>	539
<a href="#">benchmark-ftrsm-mp.C</a>	540
<a href="#">benchmark-ftrsm.C</a>	541
<a href="#">benchmark-ftrsv.C</a>	541
<a href="#">benchmark-ftrtri.C</a>	542
<a href="#">benchmark-inverse.C</a>	542
<a href="#">benchmark-lqup-mp.C</a>	543

benchmark-lqup.C	544
benchmark-pluq.C	544
benchmark-quasisep.C	545
benchmark-storage-transpose.C	546
benchmark-wino.C	547
mainpage.doxy	548
det.C	548
matmul.C	548
rank.C	549
solve.C	549
checker_charpoly.inl	549
checker_det.inl	550
checker_empty.h	550
checker_fgemm.inl	550
checker_ftrsm.inl	551
checker_invert.inl	551
checker_pluq.inl	551
checkers.doxy	552
checkers_fflas.h	552
checkers_fflas.inl	552
checkers_ffpack.h	553
checkers_ffpack.inl	553
config-blas.h	554
config.h	561
fflas-ffpack/config.h	564
fflas-ffpack-config.h	
Defaults for optimised values	567
fflas-ffpack-default-thresholds.h	568
fflas-ffpack-thresholds.h	569
fflas-ffpack.doxy	569
fflas-ffpack.h	
Includes <b>FFLAS</b> and <b>FFPACK</b>	569
fflas.doxy	569
fflas.h	
<b>Finite Field Linear Algebra Subroutines</b>	569
fflas_bounds.inl	571
fflas_enum.h	571
fflas_fadd.h	572
fflas_fadd.inl	573
fflas_fassign.h	574
fflas_fassign.inl	575
fflas_faxpy.inl	575
fflas_fdot.inl	576
fflas_fgemm.inl	577
fgemm_classical.inl	579
fgemm_classical_mp.inl	
Matrix multiplication with multiprecision input (either over $\mathbb{Z}$ or over $\mathbb{Z}/p\mathbb{Z}$ )	579
fgemm_winograd.inl	581
matmul.doxy	583
schedule_bini.inl	
Bini implementation	583
schedule_winograd.inl	583
schedule_winograd_acc.inl	584
schedule_winograd_acc_ip.inl	585
schedule_winograd_ip.inl	585
fflas_fgemv.inl	586
fflas_fgemv_mp.inl	588
fflas_fger.inl	588

fflas_fger_mp.inl	589
fflas_freduce.h	590
fflas_freduce.inl	591
fflas_freduce_mp.inl	593
fflas_freivalds.inl	593
fflas_fscal.h	594
fflas_fscal.inl	594
fflas_fscal_mp.inl	595
fflas_fsyr2k.inl	596
fflas_fsyrk.inl	597
fflas_fsyrk_strassen.inl	598
fflas_ftrmm.inl	599
fflas_ftrsm.inl	600
fflas_ftrsm_mp.inl	
Triangular system with matrix right hand side over multiprecision domain (either over $\mathbb{Z}$ or over $\mathbb{Z}/p\mathbb{Z}$ )	601
fflas_ftrsv.inl	601
fflas_helpers.inl	602
igemm.doxy	603
igemm.h	603
igemm.inl	603
igemm_kernels.h	604
igemm_kernels.inl	605
igemm_tools.h	605
igemm_tools.inl	606
fflas_level1.inl	606
fflas_level2.inl	609
fflas_level3.inl	611
fflas_pfgemm.inl	614
fflas_pftrsm.inl	615
fflas_simd.h	615
simd.doxy	617
simd128.inl	617
simd128_double.inl	618
simd128_float.inl	618
simd128_int16.inl	618
simd128_int32.inl	619
simd128_int64.inl	619
simd256.inl	620
simd256_double.inl	621
simd256_float.inl	621
simd256_int16.inl	621
simd256_int32.inl	622
simd256_int64.inl	622
simd512.inl	623
simd512_double.inl	623
simd512_float.inl	624
simd512_int32.inl	624
simd512_int64.inl	625
simd_modular.inl	625
fflas_sparse.h	625
fflas_sparse.inl	630
coo.h	632
coo_spmv.inl	632
coo_spmv.inl	633
coo_utils.inl	634
csr.h	634
csr_pspmm.inl	635

csr_pspmv.inl	636
csr_spmv.inl	636
csr_spmv.inl	638
csr_utils.inl	638
csr_hyb.h	639
csr_hyb_pspmm.inl	639
csr_hyb_pspmv.inl	640
csr_hyb_spmv.inl	641
csr_hyb_spmv.inl	641
csr_hyb_utils.inl	642
ell.h	642
ell_pspmm.inl	643
ell_pspmv.inl	643
ell_spmv.inl	644
ell_spmv.inl	645
ell_utils.inl	646
ell_simd.h	646
ell_simd_pspmv.inl	647
ell_simd_spmv.inl	648
ell_simd_utils.inl	649
hyb_zo.h	649
hyb_zo_pspmm.inl	649
hyb_zo_pspmv.inl	650
hyb_zo_spmv.inl	650
hyb_zo_spmv.inl	651
hyb_zo_utils.inl	651
read_sparse.h	652
sell.h	653
sell_pspmv.inl	653
sell_spmv.inl	654
sell_utils.inl	655
sparse_matrix_traits.h	656
utils.h	657
fflas_transpose.h	
Transpose the storage of the matrix (switch between row and col major mode)	657
ffpack.dox	659
ffpack.h	
Set of elimination based routines for dense linear algebra	659
ffpack.inl	668
ffpack_bruhatgen.inl	669
ffpack_charpoly.inl	670
ffpack_charpoly_danilevski.inl	671
ffpack_charpoly_kgfast.inl	671
ffpack_charpoly_kgfastgeneralized.inl	672
ffpack_charpoly_kglu.inl	672
ffpack_charpoly_mp.inl	673
ffpack_det_mp.inl	673
ffpack_echelonforms.inl	674
ffpack_fgesv.inl	675
ffpack_fgetrs.inl	676
ffpack_frobenius.inl	676
ffpack_fsytrf.inl	677
ffpack_ftrssyr2k.inl	678
ffpack_ftrstr.inl	679
ffpack_ftrtr.inl	679
ffpack_invert.inl	680
ffpack_krylovelim.inl	681
ffpack_ludivine.inl	681



ffpack_ludivine_mp.inl	682
ffpack_minpoly.inl	682
ffpack_permutation.inl	683
ffpack_pluq.inl	685
ffpack_pluq_mp.inl	686
ffpack_ppluq.inl	686
ffpack_rankprofiles.inl	687
field-traits.h	
Field Traits	688
field.doxy	691
rns-double-elt.h	
Rns elt structure with double support	691
rns-double-recint.inl	691
rns-double.h	
Rns structure with double support	692
rns-double.inl	693
rns-integer-mod.h	
Representation of $\mathbb{Z}/p\mathbb{Z}$ using <b>RNS</b> representation (note: fixed precision)	693
rns-integer.h	
Representation of $\mathbb{Z}$ using <b>RNS</b> representation (note: fixed precision)	694
rns.h	694
rns.inl	695
interfaces.doxy	695
fflas_c.h	695
fflas_L1_inst.C	707
fflas_L1_inst.h	708
fflas_L1_inst_implem.inl	709
fflas_L2_inst.C	710
fflas_L2_inst.h	711
fflas_L2_inst_implem.inl	712
fflas_L3_inst.C	714
fflas_L3_inst.h	715
fflas_L3_inst_implem.inl	716
fflas_lv1.C	
C functions calls for level 1 <b>FFLAS</b> in flas-c.h	717
fflas_lv2.C	
C functions calls for level 2 <b>FFLAS</b> in flas-c.h	721
fflas_lv3.C	
C functions calls for level 3 <b>FFLAS</b> in flas-c.h	726
fflas_sparse.C	
C functions calls for level 1.5 and 2.5 <b>FFLAS</b> in flas-c.h	728
ffpack.C	
C functions calls for <b>FFPACK</b> in ffpack-c.h	728
ffpack_c.h	749
ffpack_inst.C	768
ffpack_inst.h	769
ffpack_inst_implem.inl	771
blockcuts.inl	774
fflas_plevel1.h	775
kaapi_routines.inl	776
parallel.h	776
pfgemm_variants.inl	783
pfgemv.inl	784
align-allocator.h	784
args-parser.h	784
bit_manipulation.h	786
cast.h	787

debug.h	
Various utilities for debugging	787
fflas_intrinsic.h	788
fflas_io.h	788
fflas_memory.h	789
fflas_randommatrix.h	790
flimits.h	792
Matio.h	793
test-utils.h	793
timer.h	794
cblas.C	794
clapack.C	795
cuda.C	796
fblas.C	796
lapack.C	797
regression-check.C	797
test-charpoly-check.C	798
test-charpoly.C	799
test-compressQ.C	800
test-det-check.C	801
test-det.C	801
test-echelon.C	802
test-fadd.C	804
test-fdot.C	805
test-fgemm-check.C	807
test-fgemm.C	808
test-fgemv.C	810
test-fger.C	812
test-fgesv.C	814
test-finit.C	815
test-fscal.C	816
test-fsyr2k.C	818
test-fsyrrk.C	819
test-fsytrf.C	821
test-ftrmm.C	822
test-ftrmv.C	823
test-ftrsm-check.C	825
test-ftrsm.C	825
test-ftrssyr2k.C	826
test-ftrstr.C	828
test-ftrsv.C	829
test-ftrtri.C	830
test-interfaces-c.c	831
test-invert-check.C	831
test-io.C	832
test-lu.C	833
test-maxdelayeddim.C	837
test-minpoly.C	837
test-multifile1.C	838
test-multifile2.C	838
test-nullspace.C	839
test-permutations.C	840
test-pluq-check.C	841
test-quasisep.C	842
test-rankprofiles.C	843
test-rpm.C	844
test-simd.C	844
test-solve.C	848

test-storage-transpose.C	849
101-fgemm.C	849
2x2-fgemm.C	850
2x2-ftrsv.C	850
2x2-pluq.C	850
fflas-101_1.C	851
fflas-101_3.C	851
fflas_101.C	852
fflas_101_lvl1.C	852
ffpack-fgesv.C	852
ffpack-solve.C	853



# Chapter 10

## Topic Documentation

### 10.1 CHECKER

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

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

### 10.2 FFLAS-FFPACK

the [FFLAS FFPACK](#) library

#### Topics

- [FFLAS](#)  
*The C-style wrapper of BLAS for finite field linear algebra.*
- [Interfaces](#)  
*Interfaces for FFLAS-FFPACK.*

#### 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.

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.

Intefaces for FFLAS-FFPACK.

C interface in folder

See also

[libs](#)

## 10.3 Matrix Multiplication Algorithms

Matrix Multiplication (level 3) 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 Supportst SSE4.1, AVX, AVX2.

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

[Todo](#) [biblio](#)

## 10.5 FFPACK

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

### 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

fields in the FFLAS-FFPACK library

### 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\\_ftsrn](#)
- 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 Ida`)
- `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 idy)`  

$$faxpy : y \leftarrow \alpha \cdot x + y.$$
- `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, [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)

- `_t N, const typename Field::Element alpha, const typename Field::ConstElement_ptr A, const size_t`  
`_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 > al-`  
`pha, 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, type-`  
`name 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, type-`  
`name 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: **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 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: **TR**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 RandIter>  
void [frand](#) (const [Field](#) &F, RandIter &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: **TRI**angular **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 [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<typename [Field](#), typename [Simd](#) = [Simd](#)<typename [Field::Element](#)>, size\_t bs = FFLAS\_TRANSPOSE\_BLOCKSIZE, typename std::enable\_if< [Simd::template is\\_same\\_element](#)< [Field](#) >::value >::type \* = nullptr, typename std::enable\_if< bs > = 1 && bs % [Simd::vect\\_size](#) == 0, ::type \* = nullptr>  
[Field::Element\\_ptr](#) [ftranspose](#) (const [Field](#) &F, 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<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, [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)



- $freduce\ x \leftarrow ymodF.$ 
  - 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) $finit\ x \leftarrow ymodF.$
- 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) $fconvert\ x \leftarrow ymodF.$
- 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) $faxpy : y \leftarrow \alpha \cdot x + \beta \cdot y.$
- 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 `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)  
 $fscaln A \leftarrow a \cdot A.$
- 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)  
 $fscal B \leftarrow a \cdot A.$
- 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)  
 $fmove : y \leftarrow \alpha \cdot x + \beta \cdot y.$
- 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: TRIangular System solve with Vector Computes  $X \leftarrow \text{op}(A^{-1})X$*
- template `INST_OR_DECL` void `ftsm` (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)  
*ftsm: TRIangular 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: TRIangular 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)
```

$\text{fassign} : A \leftarrow B.$

**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 $\text{ta} == \text{FflaSTrans}$ 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)

```

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)

```

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: Freivalds **GE**neral Matrix Multiply Random Check.

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>	<i>C</i> 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 comput $C = \alpha A \times A^T + \beta C$ , 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
in, out	<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)
```

`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)
```

`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 ftranspose()**

```
template<typename Field, typename Simd = Simd<typename Field::Element>, size_t bs = FFLAS_↵
TRANSPOSE_BLOCKSIZE, typename std::enable_if< Simd::template is_same_element< Field >::value
>::type * = nullptr, typename std::enable_if< bs > = 1 && bs % Simd::vect_size == 0, ::type *
= nullptr>
Field::Element_ptr ftranspose (
    const Field & F,
    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) [inline]
```

**11.1.3.235 maxCardinality()**

```
template<class Field, class enable = void>
Field::Residu_t maxCardinality () [inline]
```

**11.1.3.236 maxCardinality< Givaro::Modular< int64\_t > >()**

```
template<>
uint64_t maxCardinality< Givaro::Modular< int64_t > > () [inline]
```

**11.1.3.237 maxCardinality< Givaro::Modular< int32\_t > >()**

```
template<>
uint32_t maxCardinality< Givaro::Modular< int32_t > > () [inline]
```

**11.1.3.238 minCardinality()**

```
template<class Field>
Field::Residu_t minCardinality () [inline]
```

**11.1.3.239 fflas\_delete()** [1/4]

```
template<>
void fflas_delete (
    FFPACK::rns_double_elt_ptr A) [inline]
```

**11.1.3.240 fflas\_delete()** [2/4]

```
template<>
void fflas_delete (
    FFPACK::rns_double_elt_cstptr A) [inline]
```

**11.1.3.241 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]
```

**11.1.3.242 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]
```

**11.1.3.243 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)
```

**11.1.3.244 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)
```

**11.1.3.245 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)
```

**11.1.3.246 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)
```

**11.1.3.247 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]
```

**11.1.3.248 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]
```



**11.1.3.249 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)
```

**11.1.3.250 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)
```

**11.1.3.251 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)
```

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

**Parameters**

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

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

**11.1.3.252 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)
```

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

## 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

11.1.3.253 `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

11.1.3.254 `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 <code>OtherElement</code>
$incX$	stride of $X$

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

11.1.3.255 `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

11.1.3.256 `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

**11.1.3.257 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 $X$

**11.1.3.258 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 $X$

**11.1.3.259 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 $Y$

**11.1.3.260 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 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

**11.1.3.261 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)
```

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 +/-1,0 is necessary.

**11.1.3.262 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
	$\alpha$	scalar
in	$X$	vector in $F$
	$\text{incX}$	stride of $X$
out	$Y$	vector in $F$
	$\text{incY}$	stride of $Y$

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

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

**11.1.3.263 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**

	$F$	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 $Y$

**11.1.3.264 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)
```

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

**11.1.3.265 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 $Y$

**11.1.3.266 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)
```

**11.1.3.267 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)
```

**11.1.3.268 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)
```

**11.1.3.269 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)
```



**11.1.3.270 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 $B$

**11.1.3.271 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

**11.1.3.272 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

**11.1.3.273 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

**11.1.3.274 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

**11.1.3.275 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

**11.1.3.276 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

**11.1.3.277 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

**11.1.3.278 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 $B$

**11.1.3.279 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 $A$

**11.1.3.280 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 $A$

**11.1.3.281 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 $A$

**11.1.3.282 fscl()** [10/10]

```
template INST_OR_DECL void fscl (
    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)
```

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

## 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 $B$

**11.1.3.283 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

**11.1.3.284 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>

**11.1.3.285 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>

**11.1.3.286 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

**11.1.3.287 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

**11.1.3.288 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



**11.1.3.289 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

**11.1.3.290 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 $Y$

**11.1.3.291 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 $Y$

**11.1.3.292 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

## 11.1.3.293 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.

**11.1.3.294 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

**11.1.3.295 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

## 11.1.3.296 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)
```

## 11.1.3.297 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)

```

### 11.1.3.298 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)

```

### 11.1.3.299 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>

**11.1.3.300 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]
```

**11.1.3.301 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]
```

**11.1.3.302 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]
```

**11.1.3.303 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]
```

**11.1.3.304 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]
```

**11.1.3.305 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]
```

**11.1.3.306 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]
```

**11.1.3.307 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]
```

**11.1.3.308 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]
```



**11.1.3.309 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]
```

**11.1.3.310 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]
```

**11.1.3.311 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]
```

**11.1.3.312 pfzero()**

```
template<class Field>
void pfzero (
    const Field & F,
    size_t m,
    size_t n,
    typename Field::Element_ptr C,
    size_t BS = 0)
```

**11.1.3.313 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)
```

**11.1.3.314 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]
```

**11.1.3.315 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)
```

**11.1.3.316 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)
```

**11.1.3.317 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)
```

**11.1.3.318 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)
```

**11.1.3.319 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)

```

### 11.1.3.320 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)

```

### 11.1.3.321 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)

```

### 11.1.3.322 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)

```

### 11.1.3.323 parseArguments()

```

void parseArguments (
    int argc,
    char ** argv,
    Argument * args,
    bool printDefaults = true)

```

### 11.1.3.324 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

### 11.1.3.325 writeCommandString()

```

std::ostream & writeCommandString (
    std::ostream & os,
    Argument * args,
    const char * programName = nullptr)

```

writes the values of all arguments, preceded by the programName

**11.1.3.326 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)

**11.1.3.327 preamble()**

```
void preamble (
    std::ifstream & ifs,
    FFLAS_FORMAT & format) [inline]
```

**11.1.3.328 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)

**11.1.3.329 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)

**11.1.3.330 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)

**11.1.3.331 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

**11.1.3.332 alignable()**

```
template<class Element>
bool alignable () [inline]
```

**11.1.3.333 alignable< Givaro::Integer \* >()**

```
template<>
bool alignable< Givaro::Integer * > () [inline]
```

**11.1.3.334 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]
```

**11.1.3.335 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]
```

**11.1.3.336 fflas\_new() [7/7]**

```
template<class Element>
Element * fflas_new (
    const size_t m,
    const Alignment align = Alignment::DEFAULT) [inline]
```

**11.1.3.337 fflas\_delete() [3/4]**

```
template<class Element_ptr>
void fflas_delete (
    Element_ptr A) [inline]
```



**11.1.3.338 fflas\_delete()** [4/4]

```
template<class Ptr, class ... Args>
void fflas_delete (
    Ptr p,
    Args ... args) [inline]
```

**11.1.3.339 prefetch()**

```
void prefetch (
    const int64_t * ) [inline]
```

**11.1.3.340 getTLBSize()**

```
void getTLBSize (
    int & tlb) [inline]
```

**11.1.3.341 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

**11.1.3.342 queryL1CacheSize()**

```
int queryL1CacheSize () [inline]
```

**Returns**

the size in Bytes of the L1 data cache

**11.1.3.343 queryTopLevelCacheSize()**

```
int queryTopLevelCacheSize () [inline]
```

**Returns**

the size in Bytes of the L2 or L3 cache if this later is present

**11.1.3.344 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 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)

- `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)`
- `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, FieldTrait > &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, type-  
name [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 \*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)
- 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 ?

**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

### 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

### 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 [ScaAndReduce](#) (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 [ScaAndReduce](#) (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 NeedPreAxdyReduction (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 [MatF2MatF\\_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 fgemm\_convert()**

```

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) [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]
```

**11.13.1.38 min\_types()** [5/7]

```
template<>
size_t min_types (
    const RecInt::rint< 9 > & k) [inline]
```

**11.13.1.39 min\_types()** [6/7]

```
template<>
size_t min_types (
    const RecInt::rint< 10 > & k) [inline]
```

**11.13.1.40 min\_types()** [7/7]

```
template<>
size_t min_types (
    const Givaro::Integer & k) [inline]
```

**11.13.1.41 unfit()** [1/4]

```
template<class T>
bool unfit (
    T x) [inline]
```

**11.13.1.42 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]
```

**11.13.145 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)
```

**11.13.146 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)
```

**11.13.147 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.

#### 11.13.1.48 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)

```

#### 11.13.1.49 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::MachineInt>::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::MachineInt>::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::MachineInt>::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::MachineInt>::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]
```

**11.15.1.34 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]
```

**11.15.1.35 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]
```

**11.15.1.36 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]
```

**11.15.1.37 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 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)`
- `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, 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::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 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)`
- `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, 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::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 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 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 pfsppmm (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 pfsppmm (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 pfsppmm (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 pfsppmm (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 pfsppmm (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 pfsppmm_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 pfsppmm_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 pfsppmv (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 pfsppmv (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 pfsppmv (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 pfsppmv_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 pfsppmv_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 pfsppmv_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 pfsppmv_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 pfspm (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 pfspm (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 pfspm (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]

```

**11.16.1.40 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]

```

**11.16.1.41 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]

```

**11.16.1.42 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]

```

**11.16.1.43 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 ldy,
    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]
```

**11.16.148 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]
```

**11.16.149 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]
```

**11.16.150 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]
```

**11.16.151 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]
```

**11.16.152 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]
```

**11.16.153 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]
```

**11.16.1.57 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 pfspmm()** [17/18]

```
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::UnparametricTag ) [inline]
```

**11.16.1.120 pfspmm()** [18/18]

```
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,
    uint64_t kmax) [inline]
```

**11.16.1.121 pfspmv()** [13/18]

```
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 ) [inline]
```

**11.16.1.122 pfspmv()** [14/18]

```
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 ) [inline]
```

**11.16.1.123 pfspmv()** [15/18]

```
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) [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](#)< typename [Field](#)::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](#)< typename [Field](#)::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 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_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 axpyp() [1/2]

```
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) [inline]
```

### 11.20.1.2 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,
    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  $AX = B$  or  $XA = B$ .*

- `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 ldx, 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, 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, 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 Lu_↵`  
`Tag=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 Lu_↵`  
`Tag=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 Lu_↵`  
`Tag=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=FpackTileRecursive)`
- `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 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 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 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=FpackAuto, 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 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>`  
`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  $A=L\backslash U$  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=[FfpackSlabRecursive](#))  
*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=[FfpackSlabRecursive](#))  
*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=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 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.*
- `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 Ida, 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 Ida, 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 Ida, 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 Ida, typename Field::Element\_ptr Dinv, 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 Ida, typename Field::Element\_ptr Dinv, 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 Ida, typename Field::Element\_ptr Dinv, 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 Ida, 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 Ida, typename Field::Element\_ptr Dinv, const size\_t incDinv, size\_t \*P)
- `template<class Field>`  
`size_t fsytrf_UP_RPM` (const Field &Fi, const size\_t N, typename Field::Element\_ptr A, const size\_t Ida, 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 Ida, 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 Ida, 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 Ida, 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 Ida, 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 Ida, 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 Ida, 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 Ida, 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\\_frsm](#) (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<>  
size\_t [rns\\_double\\_elt\\_ptr\\_fflas\\_const\\_cast](#) (rns\_double\_elt\_cstptr x)
- template<>  
size\_t [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 [ftrrm](#) (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](#) [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)
- 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 [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 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)  
*Random Matrix with prescribed det.*
- `template<class Field, class RandIter>`  
`Field::Element_ptr RandomLTQSMMatrixWithRankandQSorder` (`Field` &F, size\_t n, size\_t r, size\_t t, typename `Field::Element_ptr` A, size\_t lda, RandIter &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

in, out	$P1$	a LAPACK permutation of size N
	$P2$	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

$F$	base field
$Side$	decides if rows (FflasLeft) or columns (FflasRight) are permuted
$Trans$	decides if the matrix is seen as columns (FflasTrans) or rows (FflasNoTrans)
$M$	size of the elements to permute
$ibeg$	first index to consider in $P$
$iend$	last index to consider in $P$
$A$	input matrix
$lda$	leading dimension of $A$
$P$	permutation in LAPACK format
$psh$	(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 permuation
<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



## Parameters

<i>NRHS</i>	number of columns (if Side = <a href="#">FFLAS::FflasLeft</a> ) or row (if Side = <a href="#">FFLAS::FflasRight</a> ) 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**

$F$	base field
$Diag$	whether $U$ should have a unit diagonal ( $FflasUnit$ ) or not ( $FflasNoUnit$ )
$M$	matrix row dimension
$N$	matrix column dimension
$A$	input matrix
$lda$	leading dimension of $A$
$P$	the row permutation
$Q$	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)
<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  $M = Q N$

$Q_t = Q^T$  If transform=false, the matrix V is not computed. See also test-colechelon for an example of use

## Parameters

	$F$	base field
	$M$	number of rows
	$N$	number of columns
in	$A$	input matrix
	$lda$	leading dimension of $A$
	$P$	the column permutation
	$Qt$	the row position of the pivots in the echelon form
	$transform$	decides whether $V$ is computed
	$LuTag$	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 `LuTag == 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 `LuTag == FfpackTileRecursive` then  $A = [L \setminus N]$  such that the same holds with  $M = NQ^TQt = Q^T$ . If `transform=false`, the matrix  $L$  is not computed. See also `test-rowechelon` for an example of use

#### Parameters

	$F$	base field
	$M$	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**

	$F$	base field
	$M$	number of rows
	$N$	number of columns
in	$A$	input matrix
	$lda$	leading dimension of $A$
	$P$	the column permutation
	$Qt$	the row 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

**11.21.3.32 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)

**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 as a single polynomial
	<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)

**11.21.3.42 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 charp (contains the base field)
out	<i>charp</i>	the characteristic polynomial of A as a single polynomial
	<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

**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:  $(v, Av, \dots, A^{N-1}v)$

**Parameters**

	<i>F</i>	the base field
out	<i>minP</i>	the minimal polynomial of A
	<i>N</i>	order of the matrix A
in	<i>A</i>	the input matrix ( $N \times N$ )
	<i>lda</i>	leading dimension of A

**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:  $(v, Av, \dots, A^kv)$

**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**

	$F$	base field
	$M$	row dimension of the matrix
	$N$	column dimension of the matrix
in	$A$	input matrix
	$lda$	leading dimension of A
	$psH$	(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_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 <i>A</i>
out	<i>x</i>	output solution vector
	<i>incx</i>	increment of <i>x</i>
	<i>b</i>	input right hand side of the system
	<i>incb</i>	increment of <i>b</i>

**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

$P$	the permutation carrying the rank profile information
$M$	the row dimension of the initial matrix
$N$	the column dimension of the initial matrix
$R$	the rank of the initial matrix
$LSm$	the row dimension of the leading submatrix considered
$LSn$	the column dimension of the leading submatrix considered
$P$	the row permutation of the PLUQ decomposition
$Q$	the column permutation of the PLUQ decomposition
$RRP$	return the row rank profile of the leading submatrix

#### Returns

the rank of the  $LSm \times LSn$  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  $R \times N$  if UpLo = FflasUpper, else T is  $M \times R$

## 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 ?

## 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  $A=L\backslash U$  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 ?

### 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=L\U 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

	<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 echelon matrix has unit pivots (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)
	<i>P</i>	positions of the R pivots
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
	<i>LuTag</i>	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  $A=L\backslash U$  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
	$UpLo$	Lower (FflasLower) means Transformation to Column Echelon Form, Upper (FflasUpper), to Row 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)

11.21.3.75 `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

## Parameters

	<i>LuTag</i>	which factorized form (CUP/PLE if FfpackSlabRecursive, PLUQ if FfpackTileRecursive)
	<i>OnlyNonZeroVectors</i>	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

	<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 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 (how many rows/columns need to be copied)
	<i>P</i>	positions of the $R$ pivots
<i>in, out</i>	<i>A</i>	input/output matrix
	<i>lda</i>	leading dimension of $A$
	<i>LuTag</i>	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

	<i>F</i>	base field
	<i>UpLo</i>	selects Col (FflasLower) or Row (FflasUpper) 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)

#### 11.21.3.78 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 <a href="#">Field</a>
<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 <a href="#">Field</a>
<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 <a href="#">Field</a>
<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

<i>N</i>	size of the matrix
<i>r</i>	rank of the matrix
<i>P</i>	permutation matrix
<i>Q</i>	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

<i>Fi</i>	base <a href="#">Field</a>
<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>P</i>	permutation matrix
<i>Q</i>	permutation matrix
<i>A</i>	the matrix to compact
<i>lda</i>	leading dimension of A
<i>X</i>	matrix that will stock the representation
<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

**11.21.3.84 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 <a href="#">Field</a>
<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	$N$	size of the matrix
in	$R$	rank
in	$P$	permutation Matrix
in	$Q$	permutation Matrix
out	$M$	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 Compute 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 x 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
	<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

**11.21.3.91 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



## Parameters

	$MI$	a permutation matrix such that $(DI + TISI)MI$ is the L factor of the Bruhat generator
	$B$	an $N \times t$ dense matrix
	$ldb$	leading dimension of B
	$\beta$	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 > &completed← Factors, 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 ← 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

### 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>
typedef MMHelperAlgo::Winograd value
```

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

### 12.2 ALL< v > Struct Template Reference

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

### 12.3 ArbitraryPrecIntTag Struct Reference

Arbitrary precision integers: GMP.

```
#include <field-traits.h>
```

#### 12.3.1 Detailed Description

Arbitrary precision integers: GMP.

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

### 12.4 AreEqual< X, Y > Class Template Reference

```
#include <fflas_enum.h>
```

#### Static Public Attributes

- static const bool [value](#) = false

## 12.4.1 Field Documentation

### 12.4.1.1 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](#)

## 12.5 Argument Struct Reference

```
#include <args-parser.h>
```

### Data Fields

- char [c](#)
- const char \* [example](#)
- const char \* [helpString](#)
- [ArgumentType](#) type
- void \* [data](#)

### 12.5.1 Field Documentation

#### 12.5.1.1 c

```
char c
```

#### 12.5.1.2 example

```
const char* example
```

#### 12.5.1.3 helpString

```
const char* helpString
```

#### 12.5.1.4 type

```
ArgumentType type
```

#### 12.5.1.5 data

```
void* data
```

The documentation for this struct was generated from the following file:

- [args-parser.h](#)

## 12.6 array< T > Class Template Reference

STL class.

### Data Structures

- class [const\\_iterator](#)
- class [const\\_reverse\\_iterator](#)
- class [iterator](#)
- class [reverse\\_iterator](#)

**Data Fields**

- [T elements](#)  
*STL member.*

**12.6.1 Detailed Description**

STL class.

**12.6.2 Field Documentation****12.6.2.1 elements**

`T elements`

STL member.

The documentation for this class was generated from the following files:

**12.7 associatedDelayedField< Field > Struct Template Reference**

```
#include <field-traits.h>
```

**Public Types**

- typedef [Field field](#)
- typedef [Field & type](#)

**12.7.1 Member Typedef Documentation****12.7.1.1 field**

```
template<class Field>
typedef Field field
```

**12.7.1.2 type**

```
template<class Field>
typedef Field& type
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

**12.8 Auto Struct Reference**

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

**12.9 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<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`
- const size\_t m
- const size\_t n
- const size\_t iters
- const bool inplace

## 12.9.1 Member Typedef Documentation

### 12.9.1.1 Field

```
template<typename Elt>
using Field = Modular<Elt>
```

### 12.9.1.2 Elt\_ptr

```
template<typename Elt>
using Elt_ptr = typename Field::Element_ptr
```

### 12.9.1.3 Residu

```
template<typename Elt>
using Residu = typename Field::Residu_t
```

### 12.9.1.4 enable\_if\_t

```
template<typename Elt>
template<bool B, class T = void>
using enable_if_t = typename std::enable_if<B, T>::type
```

### 12.9.1.5 is\_same\_element

```
template<typename Elt>
template<typename Simd>
using is_same_element = typename Simd::template is_same_element<Field>
```

### 12.9.1.6 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>
```

### 12.9.1.7 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>
```

### 12.9.1.8 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>
```

### 12.9.1.9 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>
```

## 12.9.2 Constructor & Destructor Documentation

### 12.9.2.1 Bench()

```
template<typename Elt>
Bench (
    size_t m,
    size_t n,
    size_t iters,
    bool inplace) [inline]
```

## 12.9.3 Member Function Documentation

### 12.9.3.1 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]
```

### 12.9.3.2 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]
```

### 12.9.3.3 doBenchs()

```
template<typename Elt>
template<typename Simd = NoSimd<Elt>, enable_if_t< is_same_element< Simd >::value > * =
nullptr>
void doBenchs () [inline]
```

### 12.9.3.4 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]
```

## 12.9.4 Field Documentation

### 12.9.4.1 F

```
template<typename Elt>
Field F [protected]
```

### 12.9.4.2 m

```
template<typename Elt>
const size_t m [protected]
```

### 12.9.4.3 n

```
template<typename Elt>
const size_t n [protected]
```

### 12.9.4.4 iters

```
template<typename Elt>
const size_t iters [protected]
```

### 12.9.4.5 inplace

```
template<typename Elt>
const bool inplace [protected]
```

The documentation for this class was generated from the following file:

- [benchmark-storage-transpose.C](#)

## 12.10 Bini Struct Reference

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

## 12.11 Block Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.12 BlockTransposeSIMD< Field, Simd, > Struct Template Reference

```
#include <fflas_transpose.h>
```

### Public Member Functions

- `template<size_t_s = Simd::vect_size, lsSimdSize<_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, lsSimdSize<_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, lsSimdSize<_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 ()`

## 12.12.1 Member Function Documentation

### 12.12.1.1 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]
```

### 12.12.1.2 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]
```

### 12.12.1.3 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]
```

### 12.12.1.4 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]
```

### 12.12.1.5 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]
```

### 12.12.1.6 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]
```

### 12.12.1.7 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](#)

## 12.13 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 ↵`  
`t *Q, const FFPACK::FFPACK_LU_TAG LuTag)`

### 12.13.1 Member Function Documentation

#### 12.13.1.1 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](#)

## 12.14 CharpolyFailed Class Reference

```
#include <ffpack.h>
```



The documentation for this class was generated from the following file:

- [ffpack.h](#)

## 12.15 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)

### 12.15.1 Constructor & Destructor Documentation

#### 12.15.1.1 Checker\_Empty()

```
template<class Field>
template<typename... Params>
Checker_Empty (
    Params... parameters) [inline]
```

### 12.15.2 Member Function Documentation

#### 12.15.2.1 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](#)

## 12.16 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)

### 12.16.1 Constructor & Destructor Documentation

#### 12.16.1.1 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]
```

### 12.16.1.2 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]
```

### 12.16.1.3 ~CheckerImplem\_charpoly()

```
template<class Field, class Polynomial>
~CheckerImplem_charpoly () [inline]
```

## 12.16.2 Member Function Documentation

### 12.16.2.1 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](#)

## 12.17 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.*

## 12.17.1 Constructor & Destructor Documentation

### 12.17.1.1 CheckerImplem\_Det() [1/2]

```
template<class Field>
CheckerImplem_Det (
    const Field & F_,
    size_t n_,
    typename Field::ConstElement\_ptr A,
    size_t lda) [inline]
```

### 12.17.1.2 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]
```

### 12.17.1.3 ~CheckerImplem\_Det()

```
template<class Field>
~CheckerImplem_Det () [inline]
```

## 12.17.2 Member Function Documentation

### 12.17.2.1 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.inl](#)
- [checkers\\_ffpack.h](#)

## 12.18 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)

### 12.18.1 Constructor & Destructor Documentation

#### 12.18.1.1 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]
```

### 12.18.1.2 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]
```

### 12.18.1.3 ~CheckerImplem\_fgemm()

```
template<class Field>
~CheckerImplem_fgemm () [inline]
```

## 12.18.2 Member Function Documentation

### 12.18.2.1 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\\_fgemm.inl](#)
- [checkers\\_fflas.h](#)

## 12.19 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 ldX)

### 12.19.1 Constructor & Destructor Documentation

#### 12.19.1.1 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]
```

### 12.19.1.2 CheckerImplem\_ftsm() [2/2]

```
template<class Field>
CheckerImplem_ftsm (
    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]
```

### 12.19.1.3 ~CheckerImplem\_ftsm()

```
template<class Field>
~CheckerImplem_ftsm () [inline]
```

## 12.19.2 Member Function Documentation

### 12.19.2.1 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](#)

## 12.20 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)

### 12.20.1 Constructor & Destructor Documentation

#### 12.20.1.1 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]

```

### 12.20.1.2 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]

```

### 12.20.1.3 ~CheckerImplem\_invert()

```

template<class Field>
~CheckerImplem_invert () [inline]

```

## 12.20.2 Member Function Documentation

### 12.20.2.1 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](#)

## 12.21 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.*

### 12.21.1 Constructor & Destructor Documentation

#### 12.21.1.1 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]

```

**12.21.1.2 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]
```

**12.21.1.3 ~CheckerImplem\_PLUQ()**

```
template<class Field>
~CheckerImplem_PLUQ () [inline]
```

**12.21.2 Member Function Documentation****12.21.2.1 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](#)

**12.22 Classic Struct Reference**

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

**12.23 Column Struct Reference**

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

**12.24 CompactElement< Element > Struct Template Reference****Public Types**

- typedef Element [type](#)

## 12.24.1 Member Typedef Documentation

### 12.24.1.1 type

```
template<class Element>
typedef Element type
```

The documentation for this struct was generated from the following file:

- [test-io.C](#)

## 12.25 compatible\_data\_type< Field > Struct Template Reference

### Static Public Attributes

- static constexpr bool [value](#) = true

### 12.25.1 Field Documentation

#### 12.25.1.1 value

```
template<typename Field>
bool value = true [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [benchmark-fgemv.C](#)

## 12.26 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](#))

### 12.26.1 Constructor & Destructor Documentation

#### 12.26.1.1 Compose() [1/5]

```
template<typename H1 = Sequential, typename H2 = Sequential>
Compose () [inline]
```

#### 12.26.1.2 Compose() [2/5]

```
template<typename H1 = Sequential, typename H2 = Sequential>
Compose (
    const Compose< H1, H2 > & other) [inline]
```

#### 12.26.1.3 Compose() [3/5]

```
template<typename H1 = Sequential, typename H2 = Sequential>
Compose (
    const Sequential & S) [inline]
```



**12.26.1.4 Compose() [4/5]**

```
template<typename H1 = Sequential, typename H2 = Sequential>
Compose (
    size_t th1,
    size_t th2) [inline]
```

**12.26.1.5 Compose() [5/5]**

```
template<typename H1 = Sequential, typename H2 = Sequential>
Compose (
    const H1 & o1,
    const H2 & o2) [inline]
```

**12.26.2 Member Function Documentation****12.26.2.1 first\_component()**

```
template<typename H1 = Sequential, typename H2 = Sequential>
H1 first_component () const [inline]
```

**12.26.2.2 second\_component()**

```
template<typename H1 = Sequential, typename H2 = Sequential>
H2 second_component () const [inline]
```

**12.26.3 Friends And Related Symbol Documentation****12.26.3.1 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](#)

**12.27 array< T >::const\_iterator Class Reference**

The documentation for this class was generated from the following files:

**12.28 string::const\_iterator Class Reference**

The documentation for this class was generated from the following files:

**12.29 vector< T >::const\_iterator Class Reference**

The documentation for this class was generated from the following files:

**12.30 array< T >::const\_reverse\_iterator Class Reference**

The documentation for this class was generated from the following files:

**12.31 string::const\_reverse\_iterator Class Reference**

The documentation for this class was generated from the following files:

## 12.32 `vector< T >::const_reverse_iterator` Class Reference

The documentation for this class was generated from the following files:

## 12.33 `ConvertTo< T >` Struct Template Reference

Force conversion to appropriate element type of ElementCategory T.

```
#include <field-traits.h>
```

### 12.33.1 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::FixedPrecIntTag>](#) tries conversion of Modular<Integer> to Modular<RecInt<K>>
- [ConvertTo<ElementCategories::ArbitraryPrecIntTag>](#) tries conversion of Modular<Integer> to RNSInteger

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.34 `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

### 12.34.1 Member Typedef Documentation

#### 12.34.1.1 Self

```
template<class ValT, class IdxT>
```

```
using Self = Coo<ValT, IdxT>
```

## 12.34.2 Constructor & Destructor Documentation

### 12.34.2.1 Coo() [1/4]

```
template<class ValT, class IdxT>
Coo (
    ValT v,
    IdxT r,
    IdxT c) [inline]
```

### 12.34.2.2 Coo() [2/4]

```
template<class ValT, class IdxT>
Coo () [default]
```

### 12.34.2.3 Coo() [3/4]

```
template<class ValT, class IdxT>
Coo (
    const Self & ) [default]
```

### 12.34.2.4 Coo() [4/4]

```
template<class ValT, class IdxT>
Coo (
    Self && ) [default]
```

## 12.34.3 Member Function Documentation

### 12.34.3.1 operator=() [1/2]

```
template<class ValT, class IdxT>
Self & operator= (
    const Self & ) [default]
```

### 12.34.3.2 operator=() [2/2]

```
template<class ValT, class IdxT>
Self & operator= (
    Self && ) [default]
```

## 12.34.4 Field Documentation

### 12.34.4.1 val

```
template<class ValT, class IdxT>
ValT val = 0
```

### 12.34.4.2 row

```
template<class ValT, class IdxT>
IdxT row = 0
```

### 12.34.4.3 col

```
template<class ValT, class IdxT>
IdxT col = 0
```

The documentation for this struct was generated from the following file:

- [csr\\_hyb\\_utils.inl](#)

## 12.35 **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

## 12.35.1 Constructor & Destructor Documentation

### 12.35.1.1 **Coo()** [1/4]

```
template<class Field>
Coo () [default]
```

### 12.35.1.2 **Coo()** [2/4]

```
template<class Field>
Coo (
    typename Field::Element v,
    index_t r,
    index_t c) [inline]
```

### 12.35.1.3 **Coo()** [3/4]

```
template<class Field>
Coo (
    const Self & ) [default]
```

### 12.35.1.4 **Coo()** [4/4]

```
template<class Field>
Coo (
    Self && ) [default]
```

## 12.35.2 Member Function Documentation

### 12.35.2.1 **operator=()** [1/2]

```
template<class Field>
Self & operator= (
    const Self & ) [default]
```

### 12.35.2.2 **operator=()** [2/2]

```
template<class Field>
Self & operator= (
    Self && ) [default]
```

### 12.35.3 Field Documentation

#### 12.35.3.1 `val`

```
template<class Field>
Field::Element val = 0
```

#### 12.35.3.2 `col`

```
template<class Field>
index_t col = 0
```

#### 12.35.3.3 `row`

```
template<class Field>
index_t row = 0
```

#### 12.35.3.4 `deleted`

```
template<class Field>
bool deleted = false
```

The documentation for this struct was generated from the following file:

- [read\\_sparse.h](#)

## 12.36 `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

### 12.36.1 Member Typedef Documentation

#### 12.36.1.1 `Self`

```
template<class ValT, class IdxT>
using Self = Coo<ValT, IdxT>
```

### 12.36.2 Constructor & Destructor Documentation

#### 12.36.2.1 `Coo()` [1/4]

```
template<class ValT, class IdxT>
Coo (
    ValT v,
```

```

IdxT r,
IdxT c) [inline]

```

#### 12.36.2.2 **Coo()** [2/4]

```

template<class ValT, class IdxT>
Coo () [default]

```

#### 12.36.2.3 **Coo()** [3/4]

```

template<class ValT, class IdxT>
Coo (
    const Self & ) [default]

```

#### 12.36.2.4 **Coo()** [4/4]

```

template<class ValT, class IdxT>
Coo (
    Self && ) [default]

```

### 12.36.3 Member Function Documentation

#### 12.36.3.1 **operator=()** [1/2]

```

template<class ValT, class IdxT>
Self & operator= (
    const Self & ) [default]

```

#### 12.36.3.2 **operator=()** [2/2]

```

template<class ValT, class IdxT>
Self & operator= (
    Self && ) [default]

```

### 12.36.4 Field Documentation

#### 12.36.4.1 **val**

```

template<class ValT, class IdxT>
ValT val = 0

```

#### 12.36.4.2 **row**

```

template<class ValT, class IdxT>
IdxT row = 0

```

#### 12.36.4.3 **col**

```

template<class ValT, class IdxT>
IdxT col = 0

```

The documentation for this struct was generated from the following file:

- [sell\\_utils.inl](#)

## 12.37 **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

**12.37.1 Field Documentation****12.37.1.1 \_coo16**

```
template<class Field>
FFLAS::Sparse<Field, SparseMatrix_t::COO, int16_t>* _coo16 = nullptr
```

**12.37.1.2 \_coo32**

```
template<class Field>
FFLAS::Sparse<Field, SparseMatrix_t::COO, int32_t>* _coo32 = nullptr
```

**12.37.1.3 \_coo64**

```
template<class Field>
FFLAS::Sparse<Field, SparseMatrix_t::COO, int64_t>* _coo64 = nullptr
```

**12.37.1.4 \_coo16\_zo**

```
template<class Field>
FFLAS::Sparse<Field, SparseMatrix_t::COO_ZO, int16_t>* _coo16_zo = nullptr
```

**12.37.1.5 \_coo32\_zo**

```
template<class Field>
FFLAS::Sparse<Field, SparseMatrix_t::COO_ZO, int32_t>* _coo32_zo = nullptr
```

**12.37.1.6 \_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](#)

**12.38 count\_nonconst\_lvalue\_reference< T > Struct Template Reference**

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

**12.39 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

## 12.39.1 Field Documentation

### 12.39.1.1 [\\_csr16](#)

```
template<class Field>
FFLAS::Sparse<Field, SparseMatrix_t::CSR, int16_t>* _csr16 = nullptr
```

### 12.39.1.2 [\\_csr32](#)

```
template<class Field>
FFLAS::Sparse<Field, SparseMatrix_t::CSR, int32_t>* _csr32 = nullptr
```

### 12.39.1.3 [\\_csr64](#)

```
template<class Field>
FFLAS::Sparse<Field, SparseMatrix_t::CSR, int64_t>* _csr64 = nullptr
```

### 12.39.1.4 [\\_csr16\\_zo](#)

```
template<class Field>
FFLAS::Sparse<Field, SparseMatrix_t::CSR_ZO, int16_t>* _csr16_zo = nullptr
```

### 12.39.1.5 [\\_csr32\\_zo](#)

```
template<class Field>
FFLAS::Sparse<Field, SparseMatrix_t::CSR_ZO, int32_t>* _csr32_zo = nullptr
```

### 12.39.1.6 [\\_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](#)

## 12.40 DefaultBoundedTag Struct Reference

Use standard field operations, but keeps track of bounds on input and output.

```
#include <field-traits.h>
```

### 12.40.1 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](#)

## 12.41 DefaultTag Struct Reference

No specific mode of action: use standard field operations.

```
#include <field-traits.h>
```



### 12.41.1 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](#)

## 12.42 DelayedTag Struct Reference

Performs field operations with delayed mod reductions. Ensures result is reduced.

```
#include <field-traits.h>
```

### 12.42.1 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](#)

## 12.43 DivideAndConquer Struct Reference

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

## 12.44 ElementTraits< Element > Struct Template Reference

[ElementTraits](#).

```
#include <field-traits.h>
```

### Public Types

- typedef [ElementCategories::GenericTag](#) value

### 12.44.1 Detailed Description

```
template<class Element>
```

```
struct FFLAS::ElementTraits< Element >
```

[ElementTraits](#).

### 12.44.2 Member Typedef Documentation

#### 12.44.2.1 value

```
template<class Element>
```

```
typedef ElementCategories::GenericTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.45 ElIMat< 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

## 12.45.1 Field Documentation

### 12.45.1.1 [\\_ell16](#)

```
template<class Field>
FFLAS::Sparse<Field, SparseMatrix\_t::ELL, int16\_t>* \_ell16 = nullptr
```

### 12.45.1.2 [\\_ell32](#)

```
template<class Field>
FFLAS::Sparse<Field, SparseMatrix\_t::ELL, int32\_t>* \_ell32 = nullptr
```

### 12.45.1.3 [\\_ell64](#)

```
template<class Field>
FFLAS::Sparse<Field, SparseMatrix\_t::ELL, int64\_t>* \_ell64 = nullptr
```

### 12.45.1.4 [\\_ell16\\_zo](#)

```
template<class Field>
FFLAS::Sparse<Field, SparseMatrix\_t::ELL\_ZO, int16\_t>* \_ell16\_zo = nullptr
```

### 12.45.1.5 [\\_ell32\\_zo](#)

```
template<class Field>
FFLAS::Sparse<Field, SparseMatrix\_t::ELL\_ZO, int32\_t>* \_ell32\_zo = nullptr
```

### 12.45.1.6 [\\_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](#)

## 12.46 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](#)

### 12.46.1 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.

### 12.46.2 Constructor & Destructor Documentation

#### 12.46.2.1 Failure()

```
Failure () [inline]
```

### 12.46.3 Member Function Documentation

#### 12.46.3.1 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.

#### 12.46.3.2 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 digg 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.

#### 12.46.3.3 setErrorMessage()

```
void setErrorMessage (
    std::ostream & stream)
```

#### 12.46.3.4 print()

```
std::ostream & print (
    std::ostream & o) const [inline]
```

overload the virtual print of `LinboxError`.

## Parameters

<i>o</i>	output stream
----------	---------------

## 12.46.4 Field Documentation

### 12.46.4.1 `_errorStream`

```
std::ostream* _errorStream [protected]
```

The documentation for this class was generated from the following file:

- [debug.h](#)

## 12.47 FailureCharpolyCheck Class Reference

```
#include <checkers_ffpack.h>
```

The documentation for this class was generated from the following file:

- [checkers\\_ffpack.h](#)

## 12.48 FailureDetCheck Class Reference

```
#include <checkers_ffpack.h>
```

The documentation for this class was generated from the following file:

- [checkers\\_ffpack.h](#)

## 12.49 FailureFgemmCheck Class Reference

```
#include <checkers_fflas.h>
```

The documentation for this class was generated from the following file:

- [checkers\\_fflas.h](#)

## 12.50 FailureInvertCheck Class Reference

```
#include <checkers_ffpack.h>
```

The documentation for this class was generated from the following file:

- [checkers\\_ffpack.h](#)

## 12.51 FailurePLUQCheck Class Reference

```
#include <checkers_ffpack.h>
```

The documentation for this class was generated from the following file:

- [checkers\\_ffpack.h](#)

## 12.52 FailureTrsmCheck Class Reference

```
#include <checkers_fflas.h>
```

The documentation for this class was generated from the following file:

- [checkers\\_fflas.h](#)

## 12.53 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`

## 12.53.1 Member Typedef Documentation

### 12.53.1.1 Field

```
template<class _Field>
using Field = _Field
```

### 12.53.1.2 Element

```
template<class _Field>
using Element = typename Field::Element
```

### 12.53.1.3 simd

```
template<class _Field>
using simd = Simd<typename _Field::Element>
```

### 12.53.1.4 vect\_t

```
template<class _Field>
using vect_t = typename simd::vect_t
```

### 12.53.1.5 scalar\_t

```
template<class _Field>
using scalar_t = typename simd::scalar_t
```

## 12.53.2 Constructor & Destructor Documentation

### 12.53.2.1 FieldSimd() [1/3]

```
template<class _Field>
FieldSimd (
    const Field & f) [inline]
```

### 12.53.2.2 FieldSimd() [2/3]

```
template<class _Field>
FieldSimd (
    const Self & ) [default]
```

### 12.53.2.3 FieldSimd() [3/3]

```
template<class _Field>
FieldSimd (
    Self && ) [default]
```

## 12.53.3 Member Function Documentation

### 12.53.3.1 operator=() [1/2]

```
template<class _Field>
Self & operator= (
    const Self & ) [default]
```

### 12.53.3.2 operator=() [2/2]

```
template<class _Field>
Self & operator= (
    Self && ) [default]
```

**12.53.3.3 init() [1/2]**

```
template<class _Field>
INLINE vect_t init (
    vect_t & x,
    const vect_t a) const [inline]
```

**12.53.3.4 init() [2/2]**

```
template<class _Field>
INLINE vect_t init (
    const vect_t a) const [inline]
```

**12.53.3.5 add() [1/2]**

```
template<class _Field>
INLINE vect_t add (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline]
```

**12.53.3.6 add() [2/2]**

```
template<class _Field>
INLINE vect_t add (
    const vect_t a,
    const vect_t b) [inline]
```

**12.53.3.7 addin()**

```
template<class _Field>
INLINE vect_t addin (
    vect_t & a,
    const vect_t b) const [inline]
```

**12.53.3.8 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]
```

**12.53.3.9 add\_r() [2/2]**

```
template<class _Field>
INLINE vect_t add_r (
    const vect_t a,
    const vect_t b) const [inline]
```

**12.53.3.10 addin\_r()**

```
template<class _Field>
INLINE vect_t addin_r (
    vect_t & a,
    const vect_t b) const [inline]
```

**12.53.3.11 sub() [1/2]**

```
template<class _Field>
INLINE vect_t sub (
    vect_t & c,
    const vect_t a,
    const vect_t b) [inline]
```

**12.53.3.12 sub() [2/2]**

```
template<class _Field>
INLINE vect_t sub (
    const vect_t a,
    const vect_t b) [inline]
```

**12.53.3.13 subin()**

```
template<class _Field>
INLINE vect_t subin (
    vect_t & a,
    const vect_t b) const [inline]
```

**12.53.3.14 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]
```

**12.53.3.15 sub\_r() [2/2]**

```
template<class _Field>
INLINE vect_t sub_r (
    const vect_t a,
    const vect_t b) const [inline]
```

**12.53.3.16 subin\_r()**

```
template<class _Field>
INLINE vect_t subin_r (
    vect_t & a,
    const vect_t b) const [inline]
```

**12.53.3.17 zero() [1/2]**

```
template<class _Field>
INLINE vect_t zero (
    vect_t & x) const [inline]
```

**12.53.3.18 zero() [2/2]**

```
template<class _Field>
INLINE vect_t zero () const [inline]
```

**12.53.3.19 mod()**

```
template<class _Field>
INLINE vect_t mod (
    vect_t & c) const [inline]
```



**12.53.3.20 mul()** [1/2]

```
template<class _Field>
INLINE vect_t mul (
    vect_t & c,
    const vect_t a,
    const vect_t b) const [inline]
```

**12.53.3.21 mul()** [2/2]

```
template<class _Field>
INLINE vect_t mul (
    const vect_t a,
    const vect_t b) const [inline]
```

**12.53.3.22 mulin()**

```
template<class _Field>
INLINE vect_t mulin (
    vect_t & a,
    const vect_t b) const [inline]
```

**12.53.3.23 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]
```

**12.53.3.24 mul\_r()** [2/2]

```
template<class _Field>
INLINE vect_t mul_r (
    const vect_t a,
    const vect_t b) const [inline]
```

**12.53.3.25 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]
```

**12.53.3.26 axpy()** [2/2]

```
template<class _Field>
INLINE vect_t axpy (
    const vect_t c,
    const vect_t a,
    const vect_t b) const [inline]
```

**12.53.3.27 axpyin()**

```
template<class _Field>
INLINE vect_t axpyin (
    vect_t & c,
```

```

    const vect_t a,
    const vect_t b) const [inline]

```

#### 12.53.3.28 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]

```

#### 12.53.3.29 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]

```

#### 12.53.3.30 axpyin\_r()

```

template<class _Field>
INLINE vect_t axpyin_r (
    vect_t & c,
    const vect_t a,
    const vect_t b) const [inline]

```

#### 12.53.3.31 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]

```

#### 12.53.3.32 maxpy() [2/2]

```

template<class _Field>
INLINE vect_t maxpy (
    const vect_t c,
    const vect_t a,
    const vect_t b) const [inline]

```

#### 12.53.3.33 maxpyin()

```

template<class _Field>
INLINE vect_t maxpyin (
    vect_t & c,
    const vect_t a,
    const vect_t b) const [inline]

```

### 12.53.4 Field Documentation

#### 12.53.4.1 vect\_size

```

template<class _Field>
const constexpr size_t vect_size = simd::vect_size [static], [constexpr]

```

### 12.53.4.2 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](#)

## 12.54 FieldTraits< Field > Struct Template Reference

FieldTrait.

```
#include <field-traits.h>
```

### Public Types

- typedef [FieldCategories::GenericTag](#) category

### Static Public Attributes

- static const bool [balanced](#) = false

### 12.54.1 Detailed Description

```
template<class Field>
```

```
struct FFLAS::FieldTraits< Field >
```

FieldTrait.

### 12.54.2 Member Typedef Documentation

#### 12.54.2.1 category

```
template<class Field>
```

```
typedef FieldCategories::GenericTag category
```

### 12.54.3 Field Documentation

#### 12.54.3.1 balanced

```
template<class Field>
```

```
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.55 Fixed Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.56 FixedPrecIntTag Struct Reference

Fixed precision integers above machine precision: Givaro::reclnt.

```
#include <field-traits.h>
```

### 12.56.1 Detailed Description

Fixed precision integers above machine precision: Givaro::reclnt.

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.57 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](#)

### 12.57.1 Constructor & Destructor Documentation

#### 12.57.1.1 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]
```

#### 12.57.1.2 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]
```

### 12.57.2 Member Function Documentation

#### 12.57.2.1 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]
```

### 12.57.2.2 initialize()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t initialize () [inline]
```

### 12.57.2.3 isTerminated()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
bool isTerminated () const [inline]
```

### 12.57.2.4 begin()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t begin () const [inline]
```

### 12.57.2.5 end()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t end () const [inline]
```

### 12.57.2.6 numblocks()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t numblocks () const [inline]
```

### 12.57.2.7 blockindex()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t blockindex () const [inline]
```

### 12.57.2.8 operator++()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t operator++ () [inline]
```

## 12.57.3 Field Documentation

### 12.57.3.1 ibeg

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t ibeg [protected]
```

### 12.57.3.2 iend

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t iend [protected]
```

### 12.57.3.3 current

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t current [protected]
```

### 12.57.3.4 firstBlockSize

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t firstBlockSize [protected]
```

### 12.57.3.5 lastBlockSize

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t lastBlockSize [protected]
```

### 12.57.3.6 changeBS

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t changeBS [protected]
```

### 12.57.3.7 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](#)

## 12.58 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 [\\_iend](#)
- 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)

## 12.58.1 Constructor & Destructor Documentation

### 12.58.1.1 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]
```

## 12.58.2 Member Function Documentation

### 12.58.2.1 initialize()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t initialize () [inline]
```

### 12.58.2.2 isTerminated()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
bool isTerminated () const [inline]
```

### 12.58.2.3 ibegin()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t ibegin () const [inline]
```

### 12.58.2.4 jbegin()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t jbegin () const [inline]
```

### 12.58.2.5 iend()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t iend () const [inline]
```

### 12.58.2.6 jend()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t jend () const [inline]
```

### 12.58.2.7 operator++()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t operator++ () [inline]
```

### 12.58.2.8 rownumblocks()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t rownumblocks () const [inline]
```

### 12.58.2.9 colnumblocks()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t colnumblocks () const [inline]
```

### 12.58.2.10 blockindex()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t blockindex () const [inline]
```

### 12.58.2.11 rowblockindex()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t rowblockindex () const [inline]
```

### 12.58.2.12 colblockindex()

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t colblockindex () const [inline]
```

## 12.58.3 Friends And Related Symbol Documentation

### 12.58.3.1 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]
```

## 12.58.4 Field Documentation

### 12.58.4.1 \_ibeg

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t _ibeg [protected]
```



#### 12.58.4.2 \_iend

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t _iend [protected]
```

#### 12.58.4.3 \_jbeg

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t _jbeg [protected]
```

#### 12.58.4.4 \_jend

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t _jend [protected]
```

#### 12.58.4.5 rowBlockSize

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t rowBlockSize [protected]
```

#### 12.58.4.6 colBlockSize

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t colBlockSize [protected]
```

#### 12.58.4.7 current

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t current [protected]
```

#### 12.58.4.8 lastRBS

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t lastRBS [protected]
```

#### 12.58.4.9 lastCBS

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t lastCBS [protected]
```

#### 12.58.4.10 changeRBS

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t changeRBS [protected]
```

#### 12.58.4.11 changeCBS

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t changeCBS [protected]
```

**12.58.4.12 numRowsBlock**

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t numRowsBlock [protected]
```

**12.58.4.13 numColBlock**

```
template<typename blocksize_t = size_t, typename Cut = CuttingStrategy::Block, typename Param
= StrategyParameter::Threads>
blocksize_t numColBlock [protected]
```

**12.58.4.14 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](#)

**12.59 ftrmmLeftLowerNoTransNonUnit< Element > Class Template Reference**

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

**12.60 ftrmmLeftLowerNoTransUnit< Element > Class Template Reference**

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

**12.61 ftrmmLeftLowerTransNonUnit< Element > Class Template Reference**

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

**12.62 ftrmmLeftLowerTransUnit< Element > Class Template Reference**

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

**12.63 ftrmmLeftUpperNoTransNonUnit< Element > Class Template Reference**

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.64 ftrmmLeftUpperNoTransUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.65 ftrmmLeftUpperTransNonUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.66 ftrmmLeftUpperTransUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.67 ftrmmRightLowerNoTransNonUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.68 ftrmmRightLowerNoTransUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.69 ftrmmRightLowerTransNonUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.70 ftrmmRightLowerTransUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.71 ftrmmRightUpperNoTransNonUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.72 `ftrmmRightUpperNoTransUnit< Element >` Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.73 `ftrmmRightUpperTransNonUnit< Element >` Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.74 `ftrmmRightUpperTransUnit< Element >` Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.75 `ftrsmLeftLowerNoTransNonUnit< Element >` Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.76 `ftrsmLeftLowerNoTransUnit< Element >` Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.77 `ftrsmLeftLowerTransNonUnit< Element >` Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.78 `ftrsmLeftLowerTransUnit< Element >` Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.79 `ftrsmLeftUpperNoTransNonUnit< Element >` Class Template Reference

Computes the maximal size for delaying the modular reduction in a triangular system resolution.

### 12.79.1 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  $\mathbb{Z}$  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](#)

## 12.80 ftrsmLeftUpperNoTransUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.81 ftrsmLeftUpperTransNonUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.82 ftrsmLeftUpperTransUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.83 ftrsmRightLowerNoTransNonUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.84 ftrsmRightLowerNoTransUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.85 `ftrsmRightLowerTransNonUnit< Element >` Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.86 `ftrsmRightLowerTransUnit< Element >` Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.87 `ftrsmRightUpperNoTransNonUnit< Element >` Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.88 `ftrsmRightUpperNoTransUnit< Element >` Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.89 `ftrsmRightUpperTransNonUnit< Element >` Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.90 `ftrsmRightUpperTransUnit< Element >` Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.91 `GenericTag` Struct Reference

default is generic

```
#include <field-traits.h>
```

### 12.91.1 Detailed Description

default is generic

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.92 GenericTag Struct Reference

generic ring.

```
#include <field-traits.h>
```

### 12.92.1 Detailed Description

generic ring.

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.93 Grain Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.94 has\_minus\_eq\_impl< C > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

### Static Public Attributes

- static constexpr bool [value](#) = type::value

### 12.94.1 Field Documentation

#### 12.94.1.1 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](#)

## 12.95 has\_minus\_impl< C > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

### Static Public Attributes

- static constexpr bool [value](#) = type::value

### 12.95.1 Field Documentation

#### 12.95.1.1 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](#)

## 12.96 has\_mul\_eq\_impl< C > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

**Static Public Attributes**

- static constexpr bool [value](#) = type::value

**12.96.1 Field Documentation****12.96.1.1 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](#)

**12.97 has\_mul\_impl< C > Struct Template Reference**

```
#include <sparse_matrix_traits.h>
```

**Static Public Attributes**

- static constexpr bool [value](#) = type::value

**12.97.1 Field Documentation****12.97.1.1 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](#)

**12.98 has\_operation< T > Struct Template Reference**

```
#include <sparse_matrix_traits.h>
```

**Static Public Attributes**

- static constexpr bool [value](#)

**12.98.1 Field Documentation****12.98.1.1 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](#)

**12.99 has\_plus\_eq\_impl< C > Struct Template Reference**

```
#include <sparse_matrix_traits.h>
```

**Static Public Attributes**

- static constexpr bool [value](#) = type::value



### 12.99.1 Field Documentation

#### 12.99.1.1 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](#)

## 12.100 has\_plus\_impl< C > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

### Static Public Attributes

- static constexpr bool [value](#) = type::value

### 12.100.1 Field Documentation

#### 12.100.1.1 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](#)

## 12.101 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

### 12.101.1 Field Documentation

#### 12.101.1.1 none

```
uint64_t none = 0_ui64 [static], [constexpr]
```

#### 12.101.1.2 coo

```
uint64_t coo = 1_ui64 [static], [constexpr]
```

#### 12.101.1.3 csr

```
uint64_t csr = 1_ui64 << 1 [static], [constexpr]
```

#### 12.101.1.4 ell

```
uint64_t ell = 1_ui64 << 2 [static], [constexpr]
```

#### 12.101.1.5 aut

```
uint64_t aut = 1_ui64 << 32 [static], [constexpr]
```

#### 12.101.1.6 pm1

```
uint64_t pm1 = 1_ui64 << 33 [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [fflas\\_sparse.h](#)

### 12.102 HelperMod< Field, ElementTraits > Struct Template Reference

The documentation for this struct was generated from the following file:

- [fflas\\_freduce.inl](#)

### 12.103 Hybrid Struct Reference

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

### 12.104 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

#### 12.104.1 Constructor & Destructor Documentation

##### 12.104.1.1 Info() [1/4]

```
Info (
    uint64_t it,
    uint64_t s,
    uint64_t p) [inline]
```

##### 12.104.1.2 Info() [2/4]

```
Info () [default]
```

##### 12.104.1.3 Info() [3/4]

```
Info (
    const Info & ) [default]
```

#### 12.104.1.4 Info() [4/4]

```
Info (
    Info && ) [default]
```

### 12.104.2 Member Function Documentation

#### 12.104.2.1 operator=() [1/2]

```
Info & operator= (
    const Info & ) [default]
```

#### 12.104.2.2 operator=() [2/2]

```
Info & operator= (
    Info && ) [default]
```

### 12.104.3 Field Documentation

#### 12.104.3.1 size

```
uint64_t size = 0
```

#### 12.104.3.2 perm

```
uint64_t perm = 0
```

#### 12.104.3.3 begin

```
uint64_t begin = 0
```

The documentation for this struct was generated from the following file:

- [csr\\_hyb\\_utils.inl](#)

## 12.105 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

### 12.105.1 Constructor & Destructor Documentation

#### 12.105.1.1 Info() [1/4]

```
Info (
    uint64_t it,
    uint64_t s,
    uint64_t p) [inline]
```

**12.105.1.2 Info() [2/4]**

```
Info () [default]
```

**12.105.1.3 Info() [3/4]**

```
Info (
    const Info & ) [default]
```

**12.105.1.4 Info() [4/4]**

```
Info (
    Info && ) [default]
```

**12.105.2 Member Function Documentation****12.105.2.1 operator=() [1/2]**

```
Info & operator= (
    const Info & ) [default]
```

**12.105.2.2 operator=() [2/2]**

```
Info & operator= (
    Info && ) [default]
```

**12.105.3 Field Documentation****12.105.3.1 size**

```
uint64_t size = 0
```

**12.105.3.2 perm**

```
uint64_t perm = 0
```

**12.105.3.3 begin**

```
uint64_t begin = 0
```

The documentation for this struct was generated from the following file:

- [sell\\_utils.inl](#)

**12.106 is\_all\_same< Args > Struct Template Reference**

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

**12.107 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

## 12.107.1 Member Typedef Documentation

### 12.107.1.1 type

```
template<class T>
using type = std::integral_constant<bool, false>
```

## 12.107.2 Field Documentation

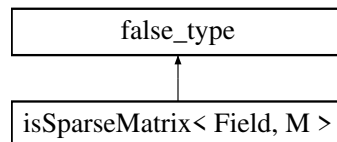
### 12.107.2.1 value

```
template<class T>
const constexpr bool value = false [static], [constexpr]
The documentation for this struct was generated from the following file:
```

- [fflas\\_simd.h](#)

## 12.108 isSparseMatrix< Field, M > Struct Template Reference

```
#include <sparse_matrix_traits.h>
Inheritance diagram for isSparseMatrix< Field, M >:
```

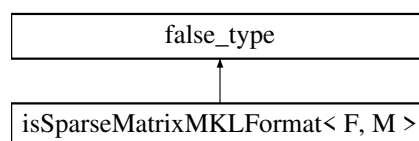


The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.109 isSparseMatrixMKLFormat< F, M > Struct Template Reference

```
#include <sparse_matrix_traits.h>
Inheritance diagram for isSparseMatrixMKLFormat< F, M >:
```

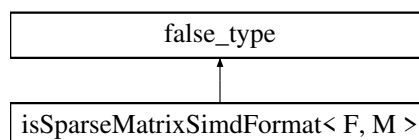


The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.110 isSparseMatrixSimdFormat< F, M > Struct Template Reference

```
#include <sparse_matrix_traits.h>
Inheritance diagram for isSparseMatrixSimdFormat< F, M >:
```



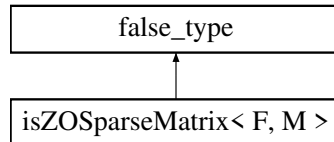
The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

### 12.111 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](#)

### 12.112 Iterative Struct Reference

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

### 12.113 array< T >::iterator Class Reference

The documentation for this class was generated from the following files:

### 12.114 string::iterator Class Reference

The documentation for this class was generated from the following files:

### 12.115 vector< T >::iterator Class Reference

The documentation for this class was generated from the following files:

### 12.116 LazyTag Struct Reference

Performs field operations with delayed mod only when necessary. Result may not be reduced.

```
#include <field-traits.h>
```

#### 12.116.1 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](#)

### 12.117 limits< T > Struct Template Reference

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.118 MachineFloatTag Struct Reference

float or double

```
#include <field-traits.h>
```

### 12.118.1 Detailed Description

float or double

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.119 MachineIntTag Struct Reference

short, int, long, long long, and unsigned variants

```
#include <field-traits.h>
```

### 12.119.1 Detailed Description

short, int, long, long long, and unsigned variants

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.120 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)

## 12.120.1 Member Typedef Documentation

### 12.120.1.1 Self\_t

```
template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
typedef MMHelper<Field,AlgoTrait,ModeTrait,ParSeqTrait> Self_t
```

### 12.120.1.2 DelayedField\_t

```
template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
typedef associatedDelayedField<constField>::type DelayedField_t
```

### 12.120.1.3 DelayedField

```
template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
typedef associatedDelayedField<constField>::field DelayedField
```

### 12.120.1.4 DFelt

```
template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
typedef DelayedField::Element DFelt
```

## 12.120.2 Constructor & Destructor Documentation

### 12.120.2.1 MMHelper() [1/5]

```
template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
MMHelper () [inline]
```



**12.120.2.2 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]
```

**12.120.2.3 MMHelper() [3/5]**

```
template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
MMHelper (
    const Field & F,
    int w,
    ParSeqTrait _PS = ParSeqTrait()) [inline]
```

**12.120.2.4 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]
```

**12.120.2.5 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]
```

**12.120.3 Member Function Documentation****12.120.3.1 initC()**

```
template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
void initC () [inline]
```

**12.120.3.2 initA()**

```
template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
void initA () [inline]
```

**12.120.3.3 initB()**

```
template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
void initB () [inline]
```

**12.120.3.4 initOut()**

```
template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
void initOut () [inline]
```

**12.120.3.5 MaxDelayedDim()**

```
template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
size_t MaxDelayedDim (
    DFelt beta) [inline]
```

**12.120.3.6 Aunfit()**

```
template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
bool Aunfit () [inline]
```

**12.120.3.7 Bunfit()**

```
template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
bool Bunfit () [inline]
```

**12.120.3.8 setOutBounds()**

```
template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
void setOutBounds (
    const size_t k,
    const DFelt alpha,
    const DFelt beta) [inline]
```

**12.120.3.9 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]
```

**12.120.3.10 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]
```

**12.120.3.11 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]
```

**12.120.3.12 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]

```

## 12.120.4 Friends And Related Symbol Documentation

### 12.120.4.1 operator<<

```

template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
std::ostream & operator<< (
    std::ostream & out,
    const Self_t & M) [friend]

```

## 12.120.5 Field Documentation

### 12.120.5.1 recLevel

```

template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
int recLevel

```

### 12.120.5.2 FieldMin

```

template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
DFElt FieldMin

```

### 12.120.5.3 FieldMax

```

template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
DFElt FieldMax

```

### 12.120.5.4 Amin

```

template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
DFElt Amin

```

### 12.120.5.5 Amax

```

template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
DFElt Amax

```

### 12.120.5.6 Bmin

```

template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
DFElt Bmin

```

### 12.120.5.7 Bmax

```

template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
DFElt Bmax

```

### 12.120.5.8 Cmin

```

template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
DFElt Cmin

```

**12.120.5.9 Cmax**

```
template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
DFelt Cmax
```

**12.120.5.10 Outmin**

```
template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
DFelt Outmin
```

**12.120.5.11 Outmax**

```
template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
DFelt Outmax
```

**12.120.5.12 MaxStorableValue**

```
template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
DFelt MaxStorableValue
```

**12.120.5.13 delayedField**

```
template<class Field, typename AlgoTrait, typename ModeTrait, typename ParSeqTrait>
const DelayedField_t delayedField
```

**12.120.5.14 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](#)

**12.121 ModeTraits< Field > Struct Template Reference**

[ModeTraits.](#)

```
#include <field-traits.h>
```

**Public Types**

- typedef [ModeCategories::DefaultTag](#) value

**12.121.1 Detailed Description**

```
template<class Field>
struct FFLAS::ModeTraits< Field >
```

[ModeTraits.](#)

**12.121.2 Member Typedef Documentation****12.121.2.1 value**

```
template<class Field>
typedef ModeCategories::DefaultTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.122 ModularBalanced< T > Class Template Reference

The documentation for this class was generated from the following file:

- [field-traits.h](#)

## 12.123 ModularBalanced< T > Class Template Reference

The documentation for this class was generated from the following file:

- [field-traits.h](#)

## 12.124 ModularTag Struct Reference

This is a modular field like e.g. `Modular<T>` or `ModularBalanced<T>`  
`#include <field-traits.h>`

### 12.124.1 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](#)

## 12.125 Montgomery< T > Class Template Reference

The documentation for this class was generated from the following file:

- [field-traits.h](#)

## 12.126 need\_field\_characteristic< Field > Struct Template Reference

### Static Public Attributes

- static constexpr bool [value](#) = false

### 12.126.1 Field Documentation

#### 12.126.1.1 value

```
template<typename Field>
bool value = false [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [benchmark-fgemv.C](#)

## 12.127 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)

## 12.127.1 Member Typedef Documentation

### 12.127.1.1 vect\_t

```
template<typename T>
using vect_t = T*
```

### 12.127.1.2 scalar\_t

```
template<typename T>
using scalar_t = T
```

### 12.127.1.3 aligned\_allocator

```
template<typename T>
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

### 12.127.1.4 aligned\_vector

```
template<typename T>
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

### 12.127.1.5 is\_same\_element

```
template<typename T>
template<class Field>
using is_same_element = std::is_same<typename Field::Element, T>
```

## 12.127.2 Member Function Documentation

### 12.127.2.1 type\_string()

```
template<typename T>
static const std::string type_string () [inline], [static]
```

### 12.127.2.2 valid()

```
template<typename T>
template<class TT>
static constexpr bool valid (
    TT p) [inline], [static], [constexpr]
```

### 12.127.2.3 compliant()

```
template<typename T>
template<class TT>
static constexpr bool compliant (
    TT n) [inline], [static], [constexpr]
```

### 12.127.3 Field Documentation

#### 12.127.3.1 vect\_size

```
template<typename T>
const constexpr size_t vect_size = 1 [static], [constexpr]
```

#### 12.127.3.2 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](#)

## 12.128 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)

### 12.128.1 Member Typedef Documentation

#### 12.128.1.1 Cut

```
template<typename C = CuttingStrategy::Block, typename P = StrategyParameter::Threads>
typedef C Cut
```

#### 12.128.1.2 Param

```
template<typename C = CuttingStrategy::Block, typename P = StrategyParameter::Threads>
typedef P Param
```

### 12.128.2 Constructor & Destructor Documentation

#### 12.128.2.1 Parallel()

```
template<typename C = CuttingStrategy::Block, typename P = StrategyParameter::Threads>
Parallel (
    size_t n = NUM\_THREADS) [inline]
```

### 12.128.3 Member Function Documentation

#### 12.128.3.1 numthreads()

```
template<typename C = CuttingStrategy::Block, typename P = StrategyParameter::Threads>
size_t numthreads () const [inline]
```

### 12.128.3.2 set\_numthreads()

```
template<typename C = CuttingStrategy::Block, typename P = StrategyParameter::Threads>
size_t & set_numthreads (
    size_t n) [inline]
```

## 12.128.4 Friends And Related Symbol Documentation

### 12.128.4.1 operator<<

```
template<typename C = CuttingStrategy::Block, typename P = StrategyParameter::Threads>
std::ostream & operator<< (
    std::ostream & out,
    const Parallel< C, P > & p) [friend]
```

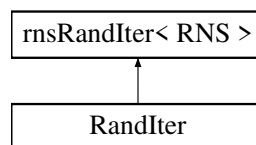
The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.129 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 assignement.*
- [RNS::Element](#) [random](#) () const
- [RNS::Element](#) & [operator\(\)](#) (typename [RNS::Element](#) &elt) const
- [RNS::Element](#) [operator\(\)](#) () const
- const [RNS](#) & [ring](#) () const

### 12.129.1 Constructor & Destructor Documentation

#### 12.129.1.1 RandIter()

```
template<typename RNS>
RandIter (
    const RNSInteger< RNS > & F,
    uint64_t seed = 0) [inline]
```

### 12.129.2 Member Function Documentation

#### 12.129.2.1 random() [1/2]

```
template<typename RNS>
RNS::Element & random (
    typename RNS::Element & elt) const [inline], [inherited]
```

[RNS](#) ring [Element](#) random assignement.

[Element](#) is supposed to be initialized



## Returns

random ring [Element](#)

## 12.129.2.2 random() [2/2]

```
template<typename RNS>
RNS::Element random () const [inline], [inherited]
```

## 12.129.2.3 operator&gt;() [1/2]

```
template<typename RNS>
RNS::Element & operator() (
    typename RNS::Element & elt) const [inline], [inherited]
```

## 12.129.2.4 operator&gt;() [2/2]

```
template<typename RNS>
RNS::Element operator() () const [inline], [inherited]
```

## 12.129.2.5 ring()

```
template<typename RNS>
const RNS & ring () const [inline], [inherited]
```

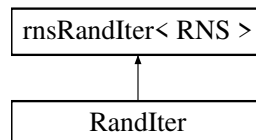
The documentation for this class was generated from the following file:

- [rns-integer.h](#)

## 12.130 RNSIntegerMod&lt; RNS &gt;::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

## 12.130.1 Constructor &amp; Destructor Documentation

## 12.130.1.1 RandIter()

```
template<typename RNS>
RandIter (
    const RNSIntegerMod< RNS > & F,
    uint64_t seed = 0) [inline]
```

## 12.130.2 Member Function Documentation

### 12.130.2.1 random() [1/2]

```
template<typename RNS>
RNS::Element & random (
    typename RNS::Element & elt) const [inline]
```

### 12.130.2.2 random() [2/2]

```
template<typename RNS>
RNS::Element random () const [inline], [inherited]
```

### 12.130.2.3 operator>() [1/2]

```
template<typename RNS>
RNS::Element & operator() (
    typename RNS::Element & elt) const [inline], [inherited]
```

### 12.130.2.4 operator>() [2/2]

```
template<typename RNS>
RNS::Element operator() () const [inline], [inherited]
```

### 12.130.2.5 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](#)

## 12.131 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)

## 12.131.1 Member Typedef Documentation

### 12.131.1.1 Element

```
template<class Field, class T>
typedef Field::Element Element
```

### 12.131.1.2 Element\_ptr

```
template<class Field, class T>
typedef Field::Element\_ptr Element\_ptr
```

## 12.131.2 Member Function Documentation

### 12.131.2.1 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](#)

## 12.132 Recursive Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.133 Recursive Struct Reference

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

## 12.134 array< T >::reverse\_iterator Class Reference

The documentation for this class was generated from the following files:

## 12.135 string::reverse\_iterator Class Reference

The documentation for this class was generated from the following files:

## 12.136 vector< T >::reverse\_iterator Class Reference

The documentation for this class was generated from the following files:

## 12.137 rint< K > Class Template Reference

The documentation for this class was generated from the following file:

- [field-traits.h](#)

## 12.138 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](#)

## 12.138.1 Member Typedef Documentation

### 12.138.1.1 integer

```
typedef Givaro::Integer integer
```

### 12.138.1.2 ModField

```
typedef Givaro::Modular<double> ModField
```

### 12.138.1.3 BasisElement

```
typedef double BasisElement
```

### 12.138.1.4 Element

```
typedef rns_double_elt Element
```

### 12.138.1.5 Element\_ptr

```
typedef rns_double_elt_ptr Element_ptr
```

### 12.138.1.6 ConstElement\_ptr

```
typedef rns_double_elt_cstptr ConstElement_ptr
```

## 12.138.2 Constructor & Destructor Documentation

### 12.138.2.1 rns\_double() [1/4]

```
rns_double (
    const integer & bound,
    size_t pbits,
    bool rnsmod = false,
    long seed = time(NULL)) [inline]
```

### 12.138.2.2 rns\_double() [2/4]

```
rns_double (
    size_t pbits,
    size_t size,
    long seed = time(NULL)) [inline]
```

### 12.138.2.3 rns\_double() [3/4]

```
template<typename Vect>
rns_double (
    const Vect & basis,
    bool rnsmod = false,
    long seed = time(NULL)) [inline]
```

### 12.138.2.4 rns\_double() [4/4]

```
rns_double (
    const RNSIntegerMod< rns_double > & basis,
    bool rnsmod = false,
    long seed = time(NULL)) [inline]
```

## 12.138.3 Member Function Documentation

### 12.138.3.1 precompute\_cst()

```
void precompute_cst (
    size_t K = 0) [inline]
```

**12.138.3.2 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]
```

**12.138.3.3 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]
```

**12.138.3.4 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]
```

**12.138.3.5 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]
```

**12.138.3.6 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]
```

**12.138.3.7 reduce()**

```
void reduce (
    size_t n,
    double * Arns,
    size_t rda,
    bool RNS_MAJOR = false) const [inline]
```

**12.138.3.8 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]
```

**12.138.3.9 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]
```

**12.138.4 Field Documentation****12.138.4.1 \_basis**

```
std::vector<double, AlignedAllocator<double, Alignment::CACHE_LINE> > _basis
```

**12.138.4.2 \_basisMax**

```
std::vector<double, AlignedAllocator<double, Alignment::CACHE_LINE> > _basisMax
```

**12.138.4.3 \_negbasis**

```
std::vector<double, AlignedAllocator<double, Alignment::CACHE_LINE> > _negbasis
```

**12.138.4.4 \_invbasis**

```
std::vector<double, AlignedAllocator<double, Alignment::CACHE_LINE> > _invbasis
```

**12.138.4.5 \_field\_rns**

```
std::vector<ModField> _field_rns
```

**12.138.4.6 \_M**

```
integer _M
```

**12.138.4.7 \_Mi**

```
std::vector<integer> _Mi
```

**12.138.4.8 \_MMi**

```
std::vector<double> _MMi
```

**12.138.4.9 \_crt\_in**

```
std::vector<double> _crt_in
```

**12.138.4.10 \_crt\_out**

```
std::vector<double> _crt_out
```

**12.138.4.11 \_size**

```
size_t _size
```

**12.138.4.12 \_pbits**

```
size_t _pbits
```

**12.138.4.13 \_ldm**

```
size_t _ldm
```

**12.138.4.14 \_mi\_sum**

```
integer _mi_sum
```

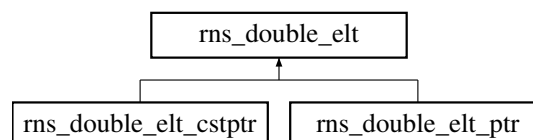
The documentation for this struct was generated from the following files:

- [rns-double.h](#)
- [rns-double-recint.inl](#)
- [rns-double.inl](#)

**12.139 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](#)

## 12.139.1 Constructor & Destructor Documentation

### 12.139.1.1 [rns\\_double\\_elt\(\)](#) [1/3]

```
rns\_double\_elt () [inline]
```

### 12.139.1.2 [~rns\\_double\\_elt\(\)](#)

```
~rns\_double\_elt () [inline]
```

### 12.139.1.3 [rns\\_double\\_elt\(\)](#) [2/3]

```
rns\_double\_elt (  
    double * p,  
    size_t r,  
    size_t a = false) [inline]
```

### 12.139.1.4 [rns\\_double\\_elt\(\)](#) [3/3]

```
rns\_double\_elt (  
    const rns\_double\_elt & x) [inline]
```

## 12.139.2 Member Function Documentation

### 12.139.2.1 [operator&\(\)](#) [1/2]

```
rns\_double\_elt\_ptr operator& () [inline]
```

### 12.139.2.2 [operator&\(\)](#) [2/2]

```
rns\_double\_elt\_cstptr operator& () const [inline]
```

## 12.139.3 Field Documentation

### 12.139.3.1 [\\_ptr](#)

```
double* \_ptr
```

### 12.139.3.2 [\\_stride](#)

```
size_t \_stride
```

### 12.139.3.3 [\\_alloc](#)

```
bool \_alloc
```

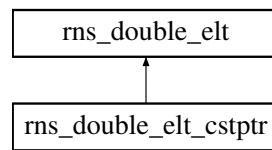
The documentation for this struct was generated from the following file:

- [rns-double-elt.h](#)

## 12.140 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](#)

## 12.140.1 Constructor & Destructor Documentation

### 12.140.1.1 rns\_double\_elt\_cstptr() [1/5]

```
rns\_double\_elt\_cstptr () [inline]
```

### 12.140.1.2 rns\_double\_elt\_cstptr() [2/5]

```
rns\_double\_elt\_cstptr (
    double * p,
    size_t r) [inline]
```

### 12.140.1.3 rns\_double\_elt\_cstptr() [3/5]

```
rns\_double\_elt\_cstptr (
    const rns\_double\_elt\_ptr & x) [inline]
```

#### 12.140.1.4 rns\_double\_elt\_cstptr() [4/5]

```
rns_double_elt_cstptr (  
    const rns_double_elt_cstptr & x) [inline]
```

#### 12.140.1.5 rns\_double\_elt\_cstptr() [5/5]

```
rns_double_elt_cstptr (  
    rns_double_elt_cstptr && ) [default]
```

### 12.140.2 Member Function Documentation

#### 12.140.2.1 operator&() [1/2]

```
rns_double_elt_cstptr * operator& () [inline]
```

#### 12.140.2.2 operator\*()

```
rns_double_elt & operator* () const [inline]
```

#### 12.140.2.3 operator[]() [1/2]

```
rns_double_elt operator[] (  
    size_t i) const [inline]
```

#### 12.140.2.4 operator[]() [2/2]

```
rns_double_elt & operator[] (  
    size_t i) [inline]
```

#### 12.140.2.5 operator++()

```
rns_double_elt_cstptr operator++ () [inline]
```

#### 12.140.2.6 operator--()

```
rns_double_elt_cstptr operator-- () [inline]
```

#### 12.140.2.7 operator+()

```
rns_double_elt_cstptr operator+ (  
    size_t inc) const [inline]
```

#### 12.140.2.8 operator-()

```
rns_double_elt_cstptr operator- (  
    size_t inc) const [inline]
```

#### 12.140.2.9 operator+=()

```
rns_double_elt_cstptr & operator+= (  
    size_t inc) [inline]
```

#### 12.140.2.10 operator-=()

```
rns_double_elt_cstptr & operator-= (  
    size_t inc) [inline]
```

**12.140.2.11 operator=()**

```
rns_double_elt_cstptr & operator= (
    const rns_double_elt_cstptr & x) [inline]
```

**12.140.2.12 operator<()**

```
bool operator< (
    const rns_double_elt_cstptr & x) [inline]
```

**12.140.2.13 operator"!=()**

```
bool operator!= (
    const rns_double_elt_cstptr & x) [inline]
```

**12.140.2.14 operator&() [2/2]**

```
rns_double_elt_cstptr operator& () const [inline], [inherited]
```

**12.140.3 Field Documentation****12.140.3.1 other**

```
rns_double_elt other
```

**12.140.3.2 \_ptr**

```
double* _ptr [inherited]
```

**12.140.3.3 \_stride**

```
size_t _stride [inherited]
```

**12.140.3.4 \_alloc**

```
bool _alloc [inherited]
```

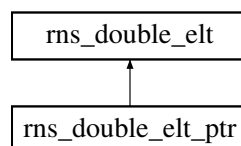
The documentation for this struct was generated from the following file:

- [rns-double-elt.h](#)

**12.141 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`

## 12.141.1 Constructor & Destructor Documentation

### 12.141.1.1 `rns_double_elt_ptr()` [1/5]

```
rns_double_elt_ptr () [inline]
```

### 12.141.1.2 `rns_double_elt_ptr()` [2/5]

```
rns_double_elt_ptr (
    double * p,
    size_t r) [inline]
```

### 12.141.1.3 `rns_double_elt_ptr()` [3/5]

```
rns_double_elt_ptr (
    const rns_double_elt_ptr & x) [inline]
```

### 12.141.1.4 `rns_double_elt_ptr()` [4/5]

```
rns_double_elt_ptr (
    const rns_double_elt_cstptr & x) [inline]
```

### 12.141.1.5 `rns_double_elt_ptr()` [5/5]

```
rns_double_elt_ptr (
    rns_double_elt_ptr && ) [default]
```

## 12.141.2 Member Function Documentation

### 12.141.2.1 `operator&()` [1/2]

```
rns_double_elt_ptr * operator& () [inline]
```

### 12.141.2.2 `operator*()`

```
rns_double_elt & operator* () [inline]
```

**12.141.2.3 operator[]() [1/2]**

```
rns_double_elt operator[] (
    size_t i) const [inline]
```

**12.141.2.4 operator[]() [2/2]**

```
rns_double_elt & operator[] (
    size_t i) [inline]
```

**12.141.2.5 operator++()**

```
rns_double_elt_ptr operator++ () [inline]
```

**12.141.2.6 operator--()**

```
rns_double_elt_ptr operator-- () [inline]
```

**12.141.2.7 operator+()**

```
rns_double_elt_ptr operator+ (
    size_t inc) [inline]
```

**12.141.2.8 operator-()**

```
rns_double_elt_ptr operator- (
    size_t inc) [inline]
```

**12.141.2.9 operator+=()**

```
rns_double_elt_ptr & operator+= (
    size_t inc) [inline]
```

**12.141.2.10 operator-=()**

```
rns_double_elt_ptr & operator-= (
    size_t inc) [inline]
```

**12.141.2.11 operator=()**

```
rns_double_elt_ptr & operator= (
    const rns_double_elt_ptr & x) [inline]
```

**12.141.2.12 operator<()**

```
bool operator< (
    const rns_double_elt_ptr & x) [inline]
```

**12.141.2.13 operator"!=()**

```
bool operator!= (
    const rns_double_elt_ptr & x) [inline]
```

**12.141.2.14 operator&() [2/2]**

```
rns_double_elt_cstptr operator& () const [inline], [inherited]
```

### 12.141.3 Field Documentation

#### 12.141.3.1 other

`rns_double_elt` other

#### 12.141.3.2 \_ptr

`double* _ptr` [inherited]

#### 12.141.3.3 \_stride

`size_t _stride` [inherited]

#### 12.141.3.4 \_alloc

`bool _alloc` [inherited]

The documentation for this struct was generated from the following file:

- [rns-double-elt.h](#)

## 12.142 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`

## 12.142.1 Member Typedef Documentation

### 12.142.1.1 integer

```
typedef Givaro::Integer integer
```

### 12.142.1.2 ModField

```
typedef Givaro::ModularExtended<double> ModField
```

### 12.142.1.3 BasisElement

```
typedef double BasisElement
```

### 12.142.1.4 Element

```
typedef rns\_double\_elt Element
```

### 12.142.1.5 Element\_ptr

```
typedef rns\_double\_elt\_ptr Element\_ptr
```

### 12.142.1.6 ConstElement\_ptr

```
typedef rns\_double\_elt\_cstptr ConstElement\_ptr
```

## 12.142.2 Constructor & Destructor Documentation

### 12.142.2.1 [rns\\_double\\_extended\(\)](#) [1/3]

```
rns\_double\_extended (
    const integer & bound,
    size_t pbits,
    bool rnsmod = false,
    long seed = time(NULL)) [inline]
```

### 12.142.2.2 [rns\\_double\\_extended\(\)](#) [2/3]

```
rns\_double\_extended (
    size_t pbits,
    size_t size,
    long seed = time(NULL)) [inline]
```

### 12.142.2.3 [rns\\_double\\_extended\(\)](#) [3/3]

```
template<typename Vect>
rns\_double\_extended (
    const Vect & basis,
    bool rnsmod = false,
    long seed = time(NULL)) [inline]
```



## 12.142.3 Member Function Documentation

### 12.142.3.1 precompute\_cst()

```
void precompute_cst () [inline]
```

### 12.142.3.2 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]
```

### 12.142.3.3 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]
```

### 12.142.3.4 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]
```

### 12.142.3.5 init() [3/3]

```
void init (  
    size_t m,  
    double * Arns,  
    const integer * A,  
    size_t lda) const [inline]
```

### 12.142.3.6 convert() [2/2]

```
void convert (  
    size_t m,  
    integer * A,  
    const double * Arns) const [inline]
```

### 12.142.3.7 reduce()

```
void reduce (  

```

```

    size_t n,
    double * Arns,
    size_t rda,
    bool RNS_MAJOR = false) const [inline]

```

## 12.142.4 Field Documentation

### 12.142.4.1 `_basis`

```
std::vector<double, AlignedAllocator<double, Alignment::CACHE_LINE> > _basis
```

### 12.142.4.2 `_basisMax`

```
std::vector<double, AlignedAllocator<double, Alignment::CACHE_LINE> > _basisMax
```

### 12.142.4.3 `_negbasis`

```
std::vector<double, AlignedAllocator<double, Alignment::CACHE_LINE> > _negbasis
```

### 12.142.4.4 `_invbasis`

```
std::vector<double, AlignedAllocator<double, Alignment::CACHE_LINE> > _invbasis
```

### 12.142.4.5 `_field_rns`

```
std::vector<ModField> _field_rns
```

### 12.142.4.6 `_M`

```
integer _M
```

### 12.142.4.7 `_Mi`

```
std::vector<integer> _Mi
```

### 12.142.4.8 `_MMi`

```
std::vector<double> _MMi
```

### 12.142.4.9 `_crt_in`

```
std::vector<double> _crt_in
```

### 12.142.4.10 `_crt_out`

```
std::vector<double> _crt_out
```

### 12.142.4.11 `_size`

```
size_t _size
```

### 12.142.4.12 `_pbits`

```
size_t _pbits
```

### 12.142.4.13 `_ldm`

```
size_t _ldm
```

The documentation for this struct was generated from the following files:

- [rns-double.h](#)
- [rns-double.inl](#)

## 12.143 RNSElementTag Struct Reference

Representation in a Residue Number System.

```
#include <field-traits.h>
```

### 12.143.1 Detailed Description

Representation in a Residue Number System.

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.144 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`

**12.144.1 Member Typedef Documentation****12.144.1.1 BasisElement**

```
template<typename RNS>
typedef RNS::BasisElement BasisElement [protected]
```

**12.144.1.2 integer**

```
template<typename RNS>
typedef Givaro::Integer integer [protected]
```

**12.144.1.3 Element**

```
template<typename RNS>
typedef RNS::Element Element
```

**12.144.1.4 Element\_ptr**

```
template<typename RNS>
typedef RNS::Element_ptr Element_ptr
```

**12.144.1.5 ConstElement\_ptr**

```
template<typename RNS>
typedef RNS::ConstElement_ptr ConstElement_ptr
```

**12.144.2 Constructor & Destructor Documentation****12.144.2.1 RNSInteger() [1/2]**

```
template<typename RNS>
RNSInteger (
    const RNS & myrns) [inline]
```

**12.144.2.2 RNSInteger() [2/2]**

```
template<typename RNS>
template<typename T>
RNSInteger (
    const T & F) [inline]
```

**12.144.3 Member Function Documentation****12.144.3.1 rns()**

```
template<typename RNS>
const RNS & rns () const [inline]
```

**12.144.3.2 size()**

```
template<typename RNS>
size_t size () const [inline]
```

**12.144.3.3 isOne()**

```
template<typename RNS>
bool isOne (
    const Element & x) const [inline]
```

**12.144.3.4 isMOne()**

```
template<typename RNS>
bool isMOne (
    const Element & x) const [inline]
```

**12.144.3.5 isZero()**

```
template<typename RNS>
bool isZero (
    const Element & x) const [inline]
```

**12.144.3.6 characteristic()**

```
template<typename RNS>
integer characteristic (
    integer & p) const [inline]
```

**12.144.3.7 cardinality()**

```
template<typename RNS>
integer cardinality (
    integer & p) const [inline]
```

**12.144.3.8 init() [1/2]**

```
template<typename RNS>
Element & init (
    Element & x) const [inline]
```

**12.144.3.9 init() [2/2]**

```
template<typename RNS>
Element & init (
    Element & x,
    const Givaro::Integer & y) const [inline]
```

**12.144.3.10 reduce() [1/2]**

```
template<typename RNS>
Element & reduce (
    Element & x,
    const Element & y) const [inline]
```

**12.144.3.11 reduce() [2/2]**

```
template<typename RNS>
Element & reduce (
    Element & x) const [inline]
```

**12.144.3.12 convert()**

```
template<typename RNS>
Givaro::Integer convert (
    Givaro::Integer & x,
    const Element & y) const [inline]
```

**12.144.3.13 assign()**

```
template<typename RNS>
Element & assign (
    Element & x,
    const Element & y) const [inline]
```

**12.144.3.14 write() [1/2]**

```
template<typename RNS>
std::ostream & write (
    std::ostream & os,
    const Element & y) const [inline]
```

**12.144.3.15 write() [2/2]**

```
template<typename RNS>
std::ostream & write (
    std::ostream & os) const [inline]
```

**12.144.4 Field Documentation****12.144.4.1 \_rns**

```
template<typename RNS>
const RNS* _rns [protected]
```

**12.144.4.2 one**

```
template<typename RNS>
Element one
```

**12.144.4.3 mOne**

```
template<typename RNS>
Element mOne
```

**12.144.4.4 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](#)

**12.145 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`

## 12.145.1 Member Typedef Documentation

### 12.145.1.1 Element

```
template<typename RNS>
typedef RNS::Element Element
```

### 12.145.1.2 Element\_ptr

```
template<typename RNS>
typedef RNS::Element\_ptr Element\_ptr
```

### 12.145.1.3 ConstElement\_ptr

```
template<typename RNS>
typedef RNS::ConstElement\_ptr ConstElement\_ptr
```

### 12.145.1.4 BasisElement

```
template<typename RNS>
typedef RNS::BasisElement BasisElement [protected]
```

### 12.145.1.5 ModField

```
template<typename RNS>
typedef Givaro::Modular<BasisElement> ModField [protected]
```

### 12.145.1.6 integer

```
template<typename RNS>
typedef Givaro::Integer integer [protected]
```

## 12.145.2 Constructor & Destructor Documentation

### 12.145.2.1 RNSIntegerMod()

```
template<typename RNS>
RNSIntegerMod (
    const integer & p,
    const RNS & myrns) [inline]
```

## 12.145.3 Member Function Documentation

### 12.145.3.1 rns()

```
template<typename RNS>
const rns\_double & rns () const [inline]
```

### 12.145.3.2 delayed()

```
template<typename RNS>
const RNSInteger< RNS > & delayed () const [inline]
```



**12.145.3.3 size()**

```
template<typename RNS>
size_t size () const [inline]
```

**12.145.3.4 isOne()**

```
template<typename RNS>
bool isOne (
    const Element & x) const [inline]
```

**12.145.3.5 isMOne()**

```
template<typename RNS>
bool isMOne (
    const Element & x) const [inline]
```

**12.145.3.6 isZero()**

```
template<typename RNS>
bool isZero (
    const Element & x) const [inline]
```

**12.145.3.7 characteristic() [1/2]**

```
template<typename RNS>
integer & characteristic (
    integer & p) const [inline]
```

**12.145.3.8 characteristic() [2/2]**

```
template<typename RNS>
integer characteristic () const [inline]
```

**12.145.3.9 cardinality() [1/2]**

```
template<typename RNS>
integer & cardinality (
    integer & p) const [inline]
```

**12.145.3.10 cardinality() [2/2]**

```
template<typename RNS>
integer cardinality () const [inline]
```

**12.145.3.11 minElement()**

```
template<typename RNS>
integer minElement () const [inline]
```

**12.145.3.12 maxElement()**

```
template<typename RNS>
integer maxElement () const [inline]
```

**12.145.3.13 init() [1/3]**

```
template<typename RNS>
Element & init (
    Element & x) const [inline]
```

**12.145.3.14 init() [2/3]**

```
template<typename RNS>
Element & init (
    Element & x,
    const Givaro::Integer & y) const [inline]
```

**12.145.3.15 reduce() [1/2]**

```
template<typename RNS>
Element & reduce (
    Element & x,
    const Element & y) const [inline]
```

**12.145.3.16 reduce() [2/2]**

```
template<typename RNS>
Element & reduce (
    Element & x) const [inline]
```

**12.145.3.17 init() [3/3]**

```
template<typename RNS>
Element & init (
    Element & x,
    const Element & y) const [inline]
```

**12.145.3.18 convert()**

```
template<typename RNS>
Givaro::Integer convert (
    Givaro::Integer & x,
    const Element & y) const [inline]
```

**12.145.3.19 assign()**

```
template<typename RNS>
Element & assign (
    Element & x,
    const Element & y) const [inline]
```

**12.145.3.20 add()**

```
template<typename RNS>
Element & add (
    Element & x,
    const Element & y,
    const Element & z) const [inline]
```

**12.145.3.21 sub()**

```
template<typename RNS>
Element & sub (
    Element & x,
    const Element & y,
    const Element & z) const [inline]
```

**12.145.3.22 neg()**

```
template<typename RNS>
Element & neg (
    Element & x,
    const Element & y) const [inline]
```

**12.145.3.23 mul()**

```
template<typename RNS>
Element & mul (
    Element & x,
    const Element & y,
    const Element & z) const [inline]
```

**12.145.3.24 axpyin()**

```
template<typename RNS>
Element & axpyin (
    Element & x,
    const Element & y,
    const Element & z) const [inline]
```

**12.145.3.25 inv()**

```
template<typename RNS>
Element & inv (
    Element & x,
    const Element & y) const [inline]
```

**12.145.3.26 areEqual()**

```
template<typename RNS>
bool areEqual (
    const Element & x,
    const Element & y) const [inline]
```

**12.145.3.27 write() [1/2]**

```
template<typename RNS>
std::ostream & write (
    std::ostream & os,
    const Element & y) const [inline]
```

**12.145.3.28 write() [2/2]**

```
template<typename RNS>
std::ostream & write (
    std::ostream & os) const [inline]
```

**12.145.3.29 reduce\_modp() [1/2]**

```
template<typename RNS>
void reduce_modp (
    size_t n,
    Element_ptr B) const [inline]
```

**12.145.3.30 write\_matrix()**

```
template<typename RNS>
std::ostream & write_matrix (
    std::ostream & c,
    const double * E,
    int n,
    int m,
    int lda) const [inline]
```

**12.145.3.31 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]
```

**12.145.3.32 reduce\_modp() [2/2]**

```
template<typename RNS>
void reduce_modp (
    size_t m,
    size_t n,
    Element_ptr B,
    size_t lda) const [inline]
```

**12.145.3.33 reduce\_modp\_rnsmajor()**

```
template<typename RNS>
void reduce_modp_rnsmajor (
    size_t n,
    Element_ptr B) const [inline]
```

**12.145.4 Field Documentation****12.145.4.1 \_p**

```
template<typename RNS>
integer _p [protected]
```

**12.145.4.2 \_Mi\_modp\_rns**

```
template<typename RNS>
std::vector<BasisElement, AlignedAllocator<BasisElement, Alignment::CACHE_LINE> > _Mi_modp_↔
rns [protected]
```

**12.145.4.3 \_iM\_modp\_rns**

```
template<typename RNS>
std::vector<BasisElement, AlignedAllocator<BasisElement, Alignment::CACHE_LINE> > _iM_modp_↔
rns [protected]
```

**12.145.4.4 \_rns**

```
template<typename RNS>
const RNS* _rns [protected]
```

**12.145.4.5** `_F`

```
template<typename RNS>
Givaro::Modular<Givaro::Integer> _F [protected]
```

**12.145.4.6** `_RNSdelayed`

```
template<typename RNS>
RNSInteger<RNS> _RNSdelayed [protected]
```

**12.145.4.7** `one`

```
template<typename RNS>
Element one
```

**12.145.4.8** `mOne`

```
template<typename RNS>
Element mOne
```

**12.145.4.9** `zero`

```
template<typename RNS>
Element zero
```

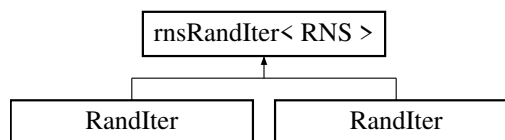
The documentation for this class was generated from the following files:

- [field-traits.h](#)
- [rns-integer-mod.h](#)
- [rns.h](#)

**12.146** `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

**12.146.1** Constructor & Destructor Documentation**12.146.1.1** `rnsRandIter()`

```
template<typename RNS>
rnsRandIter (
    const RNS & R,
    uint64_t seed = 0) [inline]
```

## 12.146.2 Member Function Documentation

### 12.146.2.1 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

### 12.146.2.2 operator>() [1/2]

```
template<typename RNS>
RNS::Element & operator() (
    typename RNS::Element & elt) const [inline]
```

### 12.146.2.3 operator>() [2/2]

```
template<typename RNS>
RNS::Element operator() () const [inline]
```

### 12.146.2.4 random() [2/2]

```
template<typename RNS>
RNS::Element random () const [inline]
```

### 12.146.2.5 ring()

```
template<typename RNS>
const RNS & ring () const [inline]
```

The documentation for this class was generated from the following file:

- [rns-double.h](#)

## 12.147 Row Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

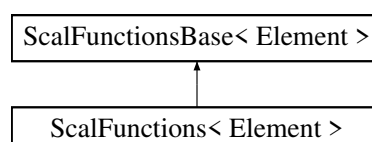
## 12.148 ruint< K > Class Template Reference

The documentation for this class was generated from the following file:

- [field-traits.h](#)

## 12.149 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)

## 12.149.1 Member Typedef Documentation

### 12.149.1.1 vectElt

```
template<typename Element>
using vectElt = vector<Element>
```

## 12.149.2 Member Function Documentation

### 12.149.2.1 genInputs()

```
template<typename Element>
static void genInputs (
    vector< vectElt > & inputs) [inline], [static]
```

#### 12.149.2.2 genInputsWithZero()

```
template<typename Element>
static void genInputsWithZero (
    vector< vectElt > & inputs) [inline], [static]
```

#### 12.149.2.3 zero()

```
template<typename Element>
static Element zero () [inline], [static]
```

#### 12.149.2.4 vand()

```
template<typename Element>
static Element vand (
    Element x1,
    Element x2) [inline], [static]
```

#### 12.149.2.5 vor()

```
template<typename Element>
static Element vor (
    Element x1,
    Element x2) [inline], [static]
```

#### 12.149.2.6 vxor()

```
template<typename Element>
static Element vxor (
    Element x1,
    Element x2) [inline], [static]
```

#### 12.149.2.7 vandnot()

```
template<typename Element>
static Element vandnot (
    Element x1,
    Element x2) [inline], [static]
```

#### 12.149.2.8 add()

```
template<typename Element>
static Element add (
    Element x1,
    Element x2) [inline], [static]
```

#### 12.149.2.9 addin()

```
template<typename Element>
static Element addin (
    Element & x1,
    Element x2) [inline], [static]
```

#### 12.149.2.10 sub()

```
template<typename Element>
static Element sub (
    Element x1,
    Element x2) [inline], [static]
```



**12.149.2.11 subin()**

```
template<typename Element>
static Element subin (
    Element & x1,
    Element x2) [inline], [static]
```

**12.149.2.12 mul()**

```
template<typename Element>
static Element mul (
    Element x1,
    Element x2) [inline], [static]
```

**12.149.2.13 mulin()**

```
template<typename Element>
static Element mulin (
    Element & x1,
    Element x2) [inline], [static]
```

**12.149.2.14 div()**

```
template<typename Element>
static Element div (
    Element x1,
    Element x2) [inline], [static]
```

**12.149.2.15 fmadd()**

```
template<typename Element>
static Element fmadd (
    Element x1,
    Element x2,
    Element x3) [inline], [static]
```

**12.149.2.16 fmaddin()**

```
template<typename Element>
static Element fmaddin (
    Element & x1,
    Element x2,
    Element x3) [inline], [static]
```

**12.149.2.17 fmsub()**

```
template<typename Element>
static Element fmsub (
    Element x1,
    Element x2,
    Element x3) [inline], [static]
```

**12.149.2.18 fmsubin()**

```
template<typename Element>
static Element fmsubin (
    Element & x1,
    Element x2,
    Element x3) [inline], [static]
```

**12.149.2.19 fnmadd()**

```
template<typename Element>
static Element fnmadd (
    Element x1,
    Element x2,
    Element x3) [inline], [static]
```

**12.149.2.20 fnmaddin()**

```
template<typename Element>
static Element fnmaddin (
    Element & x1,
    Element x2,
    Element x3) [inline], [static]
```

**12.149.2.21 lesser()**

```
template<typename Element>
static Element lesser (
    Element x1,
    Element x2) [inline], [static]
```

**12.149.2.22 lesser\_eq()**

```
template<typename Element>
static Element lesser_eq (
    Element x1,
    Element x2) [inline], [static]
```

**12.149.2.23 greater()**

```
template<typename Element>
static Element greater (
    Element x1,
    Element x2) [inline], [static]
```

**12.149.2.24 greater\_eq()**

```
template<typename Element>
static Element greater_eq (
    Element x1,
    Element x2) [inline], [static]
```

**12.149.2.25 eq()**

```
template<typename Element>
static Element eq (
    Element x1,
    Element x2) [inline], [static]
```

**12.149.2.26 unpacklo()**

```
template<typename Element>
static vectElt unpacklo (
    vectElt a,
    vectElt b) [inline], [static]
```

**12.149.2.27 unpackhi()**

```
template<typename Element>
static vectElt unpackhi (
    vectElt a,
    vectElt b) [inline], [static]
```

**12.149.2.28 unpacklohi()**

```
template<typename Element>
static void unpacklohi (
    vectElt & lo,
    vectElt & hi,
    vectElt a,
    vectElt b) [inline], [static]
```

**12.149.2.29 pack\_even()**

```
template<typename Element>
static vectElt pack_even (
    vectElt a,
    vectElt b) [inline], [static]
```

**12.149.2.30 pack\_odd()**

```
template<typename Element>
static vectElt pack_odd (
    vectElt a,
    vectElt b) [inline], [static]
```

**12.149.2.31 pack()**

```
template<typename Element>
static void pack (
    vectElt & even,
    vectElt & odd,
    vectElt a,
    vectElt b) [inline], [static]
```

**12.149.2.32 blend()**

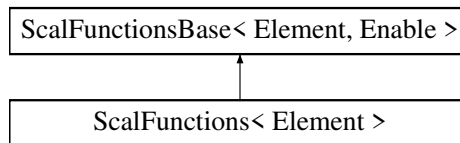
```
template<typename Element>
template<uint16_t s>
static vectElt blend (
    vectElt a,
    vectElt b) [inline], [static]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.150 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](#)

## 12.151 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](#) &)

### 12.151.1 Constructor & Destructor Documentation

#### 12.151.1.1 [Sequential](#)() [1/3]

```
Sequential () [inline]
```

#### 12.151.1.2 [Sequential](#)() [2/3]

```
Sequential (  
    size_t nth) [inline]
```

#### 12.151.1.3 [Sequential](#)() [3/3]

```
template<class Cut, class Param>  
Sequential (  
    Parallel< Cut, Param > & ) [inline]
```

### 12.151.2 Member Function Documentation

#### 12.151.2.1 [numthreads](#)()

```
size_t numthreads () const [inline]
```

### 12.151.3 Friends And Related Symbol Documentation

#### 12.151.3.1 [operator<<](#)

```
std::ostream & operator<< (  
    std::ostream & out,  
    const Sequential & ) [friend]
```

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

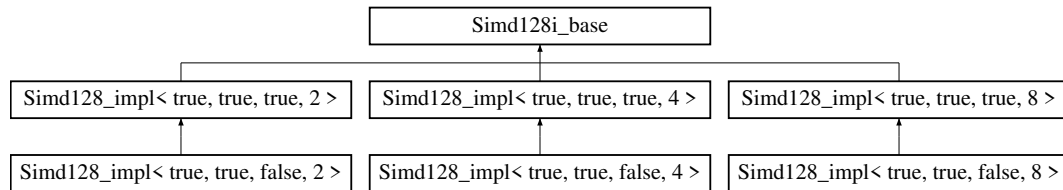
## 12.152 Simd128\_impl< ArithType, Int, Signed, Size > Struct Template Reference

The documentation for this struct was generated from the following file:

- [simd128.inl](#)

## 12.153 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)

### 12.153.1 Member Typedef Documentation

#### 12.153.1.1 vect\_t

using [vect\\_t](#) = \_\_m128i

### 12.153.2 Member Function Documentation

#### 12.153.2.1 zero()

```
static INLINE CONST vect\_t zero () [inline], [static]
```

#### 12.153.2.2 sll128()

```
template<uint8_t s>
static INLINE CONST vect\_t sll128 (
    const vect\_t a) [inline], [static]
```

#### 12.153.2.3 srl128()

```
template<uint8_t s>
static INLINE CONST vect\_t srl128 (
    const vect\_t a) [inline], [static]
```

#### 12.153.2.4 vand()

```
static INLINE CONST vect_t vand (
    const vect_t a,
    const vect_t b) [inline], [static]
```

#### 12.153.2.5 vor()

```
static INLINE CONST vect_t vor (
    const vect_t a,
    const vect_t b) [inline], [static]
```

#### 12.153.2.6 vxor()

```
static INLINE CONST vect_t vxor (
    const vect_t a,
    const vect_t b) [inline], [static]
```

#### 12.153.2.7 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](#)

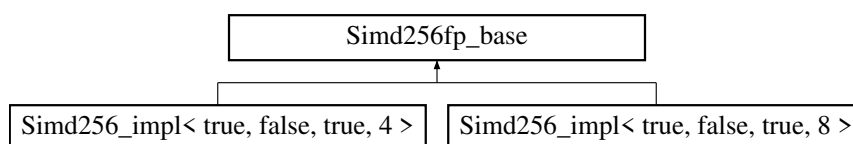
### 12.154 Simd256\_impl< ArithType, Int, Signed, Size > Struct Template Reference

The documentation for this struct was generated from the following file:

- [simd256.inl](#)

### 12.155 Simd256fp\_base Struct Reference

Inheritance diagram for Simd256fp\_base:

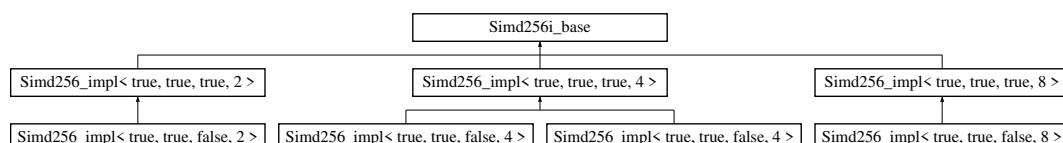


The documentation for this struct was generated from the following file:

- [simd256.inl](#)

### 12.156 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](#) ()

**12.156.1 Member Typedef Documentation****12.156.1.1 vect\_t**

using [vect\\_t](#) = \_\_m256i

**12.156.2 Member Function Documentation****12.156.2.1 zero()**

static [INLINE CONST vect\\_t zero](#) () [inline], [static]

The documentation for this struct was generated from the following file:

- [simd256.inl](#)

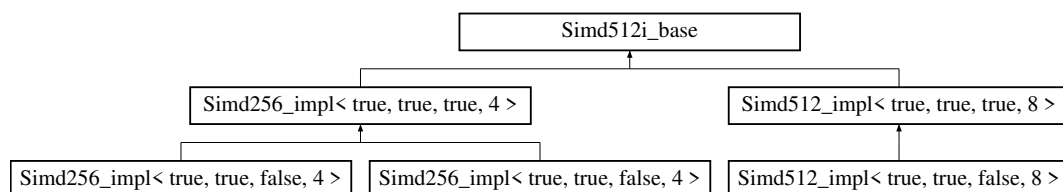
**12.157 Simd512\_impl< ArithType, Int, Signed, Size > Struct Template Reference**

The documentation for this struct was generated from the following file:

- [simd512.inl](#)

**12.158 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)

**12.158.1 Member Typedef Documentation****12.158.1.1 vect\_t**

using [vect\\_t](#) = \_\_m512i

## 12.158.2 Member Function Documentation

### 12.158.2.1 zero()

```
static INLINE CONST vect_t zero () [inline], [static]
```

### 12.158.2.2 vor()

```
static INLINE CONST vect_t vor (
    const vect_t a,
    const vect_t b) [inline], [static]
```

### 12.158.2.3 vxor()

```
static INLINE CONST vect_t vxor (
    const vect_t a,
    const vect_t b) [inline], [static]
```

### 12.158.2.4 vand()

```
static INLINE CONST vect_t vand (
    const vect_t a,
    const vect_t b) [inline], [static]
```

### 12.158.2.5 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](#)

## 12.159 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](#)

## 12.160 simdToType< T > Struct Template Reference

The documentation for this struct was generated from the following file:

- [fflas\\_simd.h](#)

## 12.161 Single Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.162 Sparse< Field, SparseMatrix\_t, IdxT, PtrT > Struct Template Reference

The documentation for this struct was generated from the following file:

- [fflas\\_sparse.h](#)



## 12.163 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

### 12.163.1 Field Documentation

#### 12.163.1.1 \_coo

```
template<class Field, int flag = HelperFlag::none>
FFLAS::CooMat<Field>* _coo = nullptr
```

#### 12.163.1.2 \_csr

```
template<class Field, int flag = HelperFlag::none>
FFLAS::CsrMat<Field>* _csr = nullptr
```

#### 12.163.1.3 \_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](#)

## 12.164 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](#)

## 12.164.1 Field Documentation

### 12.164.1.1 rowdim

uint64\_t rowdim = 0

### 12.164.1.2 coldim

uint64\_t coldim = 0

### 12.164.1.3 nOnes

uint64\_t nOnes = 0

### 12.164.1.4 nMOnes

uint64\_t nMOnes = 0

### 12.164.1.5 nOthers

uint64\_t nOthers = 0

### 12.164.1.6 nnz

uint64\_t nnz = 0

### 12.164.1.7 maxRow

uint64\_t maxRow = 0

### 12.164.1.8 minRow

uint64\_t minRow = 0

### 12.164.1.9 averageRow

uint64\_t averageRow = 0

### 12.164.1.10 deviationRow

uint64\_t deviationRow = 0

### 12.164.1.11 maxCol

uint64\_t maxCol = 0

### 12.164.1.12 minCol

uint64\_t minCol = 0

**12.164.1.13 averageCol**

```
uint64_t averageCol = 0
```

**12.164.1.14 deviationCol**

```
uint64_t deviationCol = 0
```

**12.164.1.15 minColDifference**

```
uint64_t minColDifference = 0
```

**12.164.1.16 maxColDifference**

```
uint64_t maxColDifference = 0
```

**12.164.1.17 averageColDifference**

```
uint64_t averageColDifference = 0
```

**12.164.1.18 deviationColDifference**

```
uint64_t deviationColDifference = 0
```

**12.164.1.19 minRowDifference**

```
uint64_t minRowDifference = 0
```

**12.164.1.20 maxRowDifference**

```
uint64_t maxRowDifference = 0
```

**12.164.1.21 averageRowDifference**

```
uint64_t averageRowDifference = 0
```

**12.164.1.22 deviationRowDifference**

```
uint64_t deviationRowDifference = 0
```

**12.164.1.23 nDenseRows**

```
uint64_t nDenseRows = 0
```

**12.164.1.24 nDenseCols**

```
uint64_t nDenseCols = 0
```

**12.164.1.25 nEmptyRows**

```
uint64_t nEmptyRows = 0
```

**12.164.1.26 nEmptyCols**

```
uint64_t nEmptyCols = 0
```

**12.164.1.27 nEmptyColsEnd**

```
uint64_t nEmptyColsEnd = 0
```

### 12.164.1.28 denseRows

```
std::vector<uint64_t> denseRows
```

### 12.164.1.29 denseCols

```
std::vector<uint64_t> denseCols
```

The documentation for this struct was generated from the following file:

- [utils.h](#)

## 12.165 string Class Reference

STL class.

### Data Structures

- class [const\\_iterator](#)
- class [const\\_reverse\\_iterator](#)
- class [iterator](#)
- class [reverse\\_iterator](#)

### 12.165.1 Detailed Description

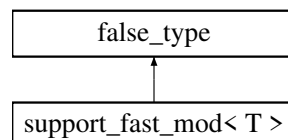
STL class.

The documentation for this class was generated from the following files:

## 12.166 support\_fast\_mod< T > Struct Template Reference

```
#include <fflas_freduce.h>
```

Inheritance diagram for support\_fast\_mod< T >:



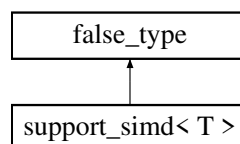
The documentation for this struct was generated from the following file:

- [fflas\\_freduce.h](#)

## 12.167 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](#)

## 12.168 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](#)

## 12.169 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](#)

## 12.170 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\\_ftranspose](#) (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`

## 12.170.1 Member Typedef Documentation

### 12.170.1.1 Field

```
template<typename Elt>
using Field = Modular<Elt>
```

### 12.170.1.2 Elt\_ptr

```
template<typename Elt>
using Elt_ptr = typename Field::Element_ptr
```

### 12.170.1.3 Residu

```
template<typename Elt>
using Residu = typename Field::Residu_t
```

### 12.170.1.4 enable\_if\_t

```
template<typename Elt>
template<bool B, class T = void>
using enable_if_t = typename std::enable_if<B, T>::type
```

### 12.170.1.5 is\_same\_element

```
template<typename Elt>
template<typename Simd>
using is_same_element = typename Simd::template is_same_element<Field>
```

### 12.170.1.6 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>
```

### 12.170.1.7 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>
```

**12.170.1.8 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>
```

**12.170.1.9 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>
```

**12.170.2 Constructor & Destructor Documentation****12.170.2.1 Test()**

```
template<typename Elt>
Test (
    size_t mm,
    size_t nn) [inline]
```

**12.170.3 Member Function Documentation****12.170.3.1 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]
```

**12.170.3.2 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]
```

**12.170.3.3 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]
```

**12.170.3.4 doTests()**

```
template<typename Elt>
template<typename Simd = NoSimd<Elt>, enable_if_t< is_same_element< Simd >::value > * =
nullptr>
bool doTests () [inline]
```

**12.170.3.5 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]
```

## 12.170.4 Field Documentation

### 12.170.4.1 F

```
template<typename Elt>
Field F [protected]
```

### 12.170.4.2 \_mm

```
template<typename Elt>
size_t _mm [protected]
```

### 12.170.4.3 \_nn

```
template<typename Elt>
size_t _nn [protected]
```

The documentation for this class was generated from the following file:

- [test-storage-transpose.C](#)

## 12.171 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](#) frame)
- 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](#)

## 12.171.1 Member Typedef Documentation

### 12.171.1.1 Element

```
template<typename Simd>
using Element = typename Simd::scalar_t
```

### 12.171.1.2 vect\_t

```
template<typename Simd>
using vect_t = typename Simd::vect_t
```

### 12.171.1.3 vectElt

```
template<typename Simd>
using vectElt = vector<Element>
```

### 12.171.1.4 enable\_if\_t

```
template<typename Simd>
template<bool B, typename T = void>
using enable_if_t = typename enable_if<B, T>::type
```

## 12.171.2 Constructor & Destructor Documentation

### 12.171.2.1 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]
```

## 12.171.3 Member Function Documentation

### 12.171.3.1 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]
```

**12.171.3.2 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]
```

**12.171.3.3 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]
```

**12.171.3.4 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]
```

**12.171.3.5 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]
```

**12.171.3.6 getStatus()**

```
template<typename Simd>
bool getStatus () const [inline]
```

**12.171.3.7 getTestName()**

```
template<typename Simd>
string getTestName () const [inline]
```

**12.171.3.8 writeResultLine()**

```
template<typename Simd>
bool writeResultLine () const [inline]
```

**12.171.3.9 writeDebugData()**

```
template<typename Simd>
void writeDebugData () const [inline]
```

**12.171.4 Field Documentation****12.171.4.1 vect\_size**

```
template<typename Simd>
size_t vect_size = Simd::vect_size [static], [constexpr]
```

**12.171.4.2 nb\_lref**

```
template<typename Simd>
size_t nb_lref [protected]
```

**12.171.4.3 name**

```
template<typename Simd>
string name [protected]
```

**12.171.4.4 inputs**

```
template<typename Simd>
vector<vectElt> inputs [protected]
```

**12.171.4.5 outputs\_simd**

```
template<typename Simd>
vector<vectElt> outputs_simd [protected]
```

**12.171.4.6 outputs\_scalar**

```
template<typename Simd>
vector<vectElt> outputs_scalar [protected]
```

The documentation for this class was generated from the following file:

- [test-simd.C](#)

**12.172 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)...))`

**12.172.1 Member Function Documentation****12.172.1.1 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](#)

**12.173 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)...))`

## 12.173.1 Member Function Documentation

### 12.173.1.1 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](#)

## 12.174 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)...))

## 12.174.1 Member Function Documentation

### 12.174.1.1 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](#)

## 12.175 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)...))

## 12.175.1 Member Function Documentation

### 12.175.1.1 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](#)

## 12.176 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)...))`

**12.176.1 Member Function Documentation****12.176.1.1 [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](#)

**12.177 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)...))`

**12.177.1 Member Function Documentation****12.177.1.1 [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](#)

**12.178 Threads Struct Reference**

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

**12.179 ThreeD Struct Reference**

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

**12.180 ThreeDAdaptive Struct Reference**

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.181 ThreeDInPlace Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.182 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](#)

### 12.182.1 Detailed Description

```
template<typename RecIterTrait = StructureHelper::Recursive, typename ParSeqTrait = ParSeqHelper::Sequential>
struct FFLAS::TRSMHelper< RecIterTrait, ParSeqTrait >
```

TRSM Helper.

### 12.182.2 Constructor & Destructor Documentation

#### 12.182.2.1 TRSMHelper() [1/3]

```
template<typename RecIterTrait = StructureHelper::Recursive, typename ParSeqTrait = ParSeqHelper::Sequential>
template<class Cut, class Param>
TRSMHelper (
    ParSeqHelper::Parallel< Cut, Param > _PS) [inline]
```

#### 12.182.2.2 TRSMHelper() [2/3]

```
template<typename RecIterTrait = StructureHelper::Recursive, typename ParSeqTrait = ParSeqHelper::Sequential>
TRSMHelper (
    ParSeqHelper::Sequential _PS) [inline]
```

#### 12.182.2.3 TRSMHelper() [3/3]

```
template<typename RecIterTrait = StructureHelper::Recursive, typename ParSeqTrait = ParSeqHelper::Sequential>
template<typename RIT, typename PST>
TRSMHelper (
    TRSMHelper< RIT, PST > &_TH) [inline]
```

### 12.182.3 Member Function Documentation

#### 12.182.3.1 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]
```

#### 12.182.3.2 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]
```

### 12.182.4 Field Documentation

#### 12.182.4.1 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](#)

## 12.183 TwoD Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.184 TwoDAdaptive Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.185 UnparametricTag Struct Reference

If the field uses a representation with infix operators.

```
#include <field-traits.h>
```

### 12.185.1 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](#)

## 12.186 `vector< T >` Class Template Reference

STL class.

### Data Structures

- class [const\\_iterator](#)
- class [const\\_reverse\\_iterator](#)
- class [iterator](#)
- class [reverse\\_iterator](#)

### Data Fields

- `T elements`  
*STL member.*

### 12.186.1 Detailed Description

STL class.

### 12.186.2 Field Documentation

#### 12.186.2.1 `elements`

`T elements`  
STL member.

The documentation for this class was generated from the following files:

## 12.187 `width< T >` Struct Template Reference

### Static Public Attributes

- static constexpr `size_t value` = `2+2*sizeof(T)`

### 12.187.1 Field Documentation

#### 12.187.1.1 `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](#)

## 12.188 Winograd Struct Reference

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

## 12.189 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 Givaro::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 [EFGFF](#)(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-ftsm-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-fttrtri.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>](#)
- class [ModularBalanced<T>](#)

## 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 dnrm2\_()

```
double dnrm2_ (
    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\_INTTYPES\_H**

```
#define HAVE_INTTYPES_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 fblas\_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, FieldTrait > &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](#), FieldTrait > &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)

### 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 type- name 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< typename Field::Element >::value &&FFLAS::support\_fast\_mod< typename Field::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< typename Field::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< typename Field::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< typename Field::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< 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, 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 [fscal](#) (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 NeedPreApxyReduction (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 NeedPreApxyReduction (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 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  $C \leftarrow \alpha \text{op}(A)B + \text{beta}C$  or  $C \leftarrow \alpha B \text{op}(A) + \text{beta}C$ .*

### 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\\_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 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 [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<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<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<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` `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)  

$$fswap: X \leftrightarrow Y.$$
  - 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, const `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, const `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 "givaroz/ring.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 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](#)>  
void [freduce](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
 $freduce A \leftarrow A mod F.$
- 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)  
 $freduce A \leftarrow B mod F.$

- 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)  
 $finit\ A \leftarrow B \bmod F.$
- 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)  
 $finit\ \text{Initializes } A \text{ in } F\mathbb{Z}.$
- 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)  
 $fconvert\ A \leftarrow B \bmod F.$
- 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)  
 $fnegin\ A \leftarrow -A.$
- 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 [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<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)  
 $faxpy : y \leftarrow \alpha \cdot x + y.$
- 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)  
 $faxpby : y \leftarrow \alpha \cdot x + \beta \cdot y.$
- 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](#)>  
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)  
 $fadd : \text{matrix addition.}$
- 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)  
 $fsub : \text{matrix subtraction.}$
- 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)  
 $fsubin\ C = C - B$
- 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, 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)
- template<class `Field`>  
void `ftsm` (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)  
*ftsm: **TR**angular **S**ystem solve with **M**atrix.*
- template<class `Field`>  
void `ftmm` (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)  
*ftmm: **TR**angular **M**atrix **M**ultiply.*
- template<class `Field`>  
void `ftmm` (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)  
*ftmm: **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 n, 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 **GE**neral **MA**trix **M**ultiply.*
- 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, \\[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 pfsppmm (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 pfsppmv (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 pfsppmv (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 pfsppmv (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 pfsppmv (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 pfsppmv (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 pfsppmv (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 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)`

## 13.137.1 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_spmml_INL`

## Functions

- `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)`

## 13.146.1 Macro Definition Documentation

### 13.146.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_spmml\_INL

```
#define __FFLASFFPACK_fflas_sparse_CSR_spmml_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-  
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::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\_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 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_spmv.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)`
- `template<typename Field, typename Simd = Simd<typename Field::Element>, size_t bs = FFLAS_TRANSPOSE_BLOCKSIZE,`  
`typename std::enable_if< Simd::template is_same_element< Field >::value >::type * = nullptr, typename std::enable_if< bs > = 1`  
`&& bs % Simd::vect_size == 0, ::type * = nullptr>`  
`Field::Element_ptr ftranspose (const Field &F, 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)`

### 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  $P1 \times \text{Diag}(I, R, P2)$  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  $AX = B$  or  $XA = B$ .*
- 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=FfpackSlabRecursive, 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 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)`
- `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 Lu←`  
`Tag=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 Lu←`  
`Tag=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 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 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 ↵`  
`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, type-`  
`name 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)`

*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, const FFPACK_CHARPOLY_TAG Charp↵`  
`Tag=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, type-`  
`name 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, 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)`

- `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, type-`  
`name Field::Element_ptr A, const size_t lda)`

- `template<class PolRing>`  
`void RandomKrylovPrecond (const PolRing &PR, std::list< typename PolRing::Element > &completed↵`  
`Factors, 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 ↵`  
`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 & 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)`
- `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  $A=L\backslash U$  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 Ida)`

*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 Ida, typename Field::Element_ptr T, const size_t Idt, 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 Ida, 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 Ida, typename Field::Element_ptr T, const size_t Idt, 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 Ida, typename Field::Element_ptr T, const size_t Idt, 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 Ida, 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 Ida, typename Field::Element_ptr T, const size_t Idt, 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 Ida, 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.*

## 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`  
`&PShHelper, 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 > &PShHelper)`

### 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 [fscal\\_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 [fscal\\_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

FflasRowMajor	row major
FflasColMajor	col major

### 13.219.2.2 FFLAS\_C\_TRANSPOSE

```
enum FFLAS_C_TRANSPOSE
```

Is matrix transposed ?

Enumerator

FflasNoTrans	Matrix is not transposed.
FflasTrans	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_implem.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 idx, 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 idx, 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, I)`
- `#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,
    P1,
    Args...)

```

**Value:**

```
P1=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](#), 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)`

## 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,`  
`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)`

## 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

<a href="#">TYPE_NONE</a>	
---------------------------	--

### Enumerator

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, type-  
name `Field::Element_ptr` A, size\_t lda)  
*Random Matrix with prescribed det.*
- `template<class Field, class RandIter>`  
`Field::Element_ptr RandomMatrixWithDet` (const `Field` &F, size\_t n, const typename `Field::Element` d, type-  
name `Field::Element_ptr` A, size\_t lda, RandIter &G)  
*Random Matrix with prescribed det.*
- `template<class Field, class RandIter>`  
`Field::Element_ptr RandomLTQSMMatrixWithRankandQSorder` (`Field` &F, size\_t n, size\_t r, size\_t t, typename  
`Field::Element_ptr` A, size\_t lda, RandIter &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\\_fscalin](#) (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\\_fscalin](#) (const [Field](#) &F, size\_t m, size\_t k, size\_t n, bool timing, uint64\_t seed)
- int [main](#) (int ac, char \*\*av)
- 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.*

## 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)

```

### 13.277.1.6 RandomMatrix()

```

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

	<i>F</i>	field
	<i>m</i>	number of rows in A
	<i>n</i>	number of cols in A
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.

## 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-fsyrrk.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_fsyrrk` (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_fsyrrk_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_fsyrrk_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\_fsyrk()

```
template<typename Field, class RandIter>
bool check_fsyrk (
    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\_fsyrk\_diag()

```
template<typename Field, class RandIter>
bool check_fsyrk_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/utils/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_ffrm (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\_ffrm()

```
template<typename Field, class RandIter>
bool check_ffrm (
    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_ftrmv (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-ffrsm-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-ffrsm.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-ftrssyr2k.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\\_fftrssyr2k](#) (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\\_fftrssyr2k\(\)](#)

```
template<typename Field, class RandIter>
bool check\_fftrssyr2k (
    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-ftrstr.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\\_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)
- 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-ffrsv.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_ffrsv (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\_ffrsv()

```
template<typename Field, class RandIter>
bool check_ffrsv (
    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\\_fftrtri](#) (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\_fftrtri()

```
template<typename Field, class RandIter>
bool check_fftrtri (
```

```

    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 [RandIter](#)>  
bool [launch\\_test](#) (const [Field](#) &F, size\_t r, size\_t m, size\_t n, [RandIter](#) &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 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].
```

Verifies that B = PLUQ where A stores [L\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/utils/args-parser.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include "fflas-ffpack/utils/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 >](#)
- class [string](#)  
*STL class.*
- class [vector< T >](#)  
*STL class.*
- class [array< T >](#)  
*STL class.*

## 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-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)

## 13.306.1 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-ftsv.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  $A \cdot x$  will not be equal to b for the later verification stage



# Index

**\_F**  
RNSIntegerMod< RNS >, [488](#)

**\_M**  
rns\_double, [467](#)  
rns\_double\_extended, [478](#)

**\_MAX\_SIZE\_MATRICES**  
benchmark-checkers.C, [525](#)

**\_MMi**  
rns\_double, [468](#)  
rns\_double\_extended, [478](#)

**\_Mi**  
rns\_double, [467](#)  
rns\_double\_extended, [478](#)

**\_Mi\_modp\_rns**  
RNSIntegerMod< RNS >, [488](#)

**\_NR\_TESTS**  
benchmark-checkers.C, [525](#)

**\_PLUQ**  
FFPACK, [361](#)

**\_RNSdelayed**  
RNSIntegerMod< RNS >, [489](#)

**\_TEST\_ONE**  
test-simd.C, [846](#)

**\_\_FFLASFFPACK\_ARITHPROG\_THRESHOLD**  
fflas-ffpack-default-thresholds.h, [569](#)

**\_\_FFLASFFPACK\_CACHE\_LINE\_SIZE**  
fflas\_sparse.h, [629](#)

**\_\_FFLASFFPACK\_CHARPOLY\_Danilevskii\_LUKrylov\_THRESHOLD**  
fflas-ffpack-default-thresholds.h, [568](#)

**\_\_FFLASFFPACK\_CHARPOLY\_LUKrylov\_ArithProg\_THRESHOLD**  
fflas-ffpack-default-thresholds.h, [568](#)

**\_\_FFLASFFPACK\_CONFIGURATION**  
cblas.C, [795](#)  
clapack.C, [795](#)  
fblas.C, [796](#)  
lapack.C, [797](#)

**\_\_FFLASFFPACK\_DIMKPENALTY**  
fflas\_pfgemm.inl, [614](#)

**\_\_FFLASFFPACK\_FORCE\_SEQ**  
benchmark-charpoly-mp.C, [523](#)

**\_\_FFLASFFPACK\_FSYRK\_THRESHOLD**  
fflas-ffpack-default-thresholds.h, [569](#)

**\_\_FFLASFFPACK\_FSYTRF\_THRESHOLD**  
fflas-ffpack-default-thresholds.h, [569](#)

**\_\_FFLASFFPACK\_FTRSSYR2K\_THRESHOLD**  
ffpack.h, [667](#)

**\_\_FFLASFFPACK\_FTRSTR\_THRESHOLD**  
ffpack.h, [667](#)

**\_\_FFLASFFPACK\_FTRTRI\_THRESHOLD**  
fflas-ffpack-default-thresholds.h, [569](#)

**\_\_FFLASFFPACK\_GAUSSJORDAN\_BASECASE**  
ffpack\_echelonforms.inl, [675](#)  
test-echelon.C, [803](#)

**\_\_FFLASFFPACK\_HAVE\_BLAS**  
config.h, [565](#)

**\_\_FFLASFFPACK\_HAVE\_CBLAS**  
cblas.C, [795](#)  
config.h, [565](#)

**\_\_FFLASFFPACK\_HAVE\_CLAPACK**  
clapack.C, [795](#)

**\_\_FFLASFFPACK\_HAVE\_CXX11**  
config.h, [565](#)

**\_\_FFLASFFPACK\_HAVE\_DGETRF**  
benchmark-dgetrf.C, [527](#)

**\_\_FFLASFFPACK\_HAVE\_DLFCN\_H**  
config.h, [565](#)

**\_\_FFLASFFPACK\_HAVE\_DTRTRI**  
benchmark-dtrtri.C, [529](#)

**\_\_FFLASFFPACK\_HAVE\_FLOAT\_H**  
config.h, [565](#)

**\_\_FFLASFFPACK\_HAVE\_INT128**  
config.h, [565](#)

**\_\_FFLASFFPACK\_HAVE\_INTPTR\_T**  
config.h, [565](#)

**\_\_FFLASFFPACK\_HAVE\_INTPTR\_T\_H**  
config.h, [565](#)

**\_\_FFLASFFPACK\_HAVE\_LAPACK**  
clapack.C, [795](#)

**\_\_FFLASFFPACK\_HAVE\_LIMITS\_H**  
config.h, [565](#)

**\_\_FFLASFFPACK\_HAVE\_LITTLE\_ENDIAN**  
config.h, [565](#)

**\_\_FFLASFFPACK\_HAVE\_PTHREAD\_H**  
config.h, [565](#)

**\_\_FFLASFFPACK\_HAVE\_STDDEF\_H**  
config.h, [565](#)

**\_\_FFLASFFPACK\_HAVE\_STDINT\_H**  
config.h, [566](#)

**\_\_FFLASFFPACK\_HAVE\_STDIO\_H**  
config.h, [566](#)

**\_\_FFLASFFPACK\_HAVE\_STDLIB\_H**  
config.h, [566](#)

**\_\_FFLASFFPACK\_HAVE\_STRINGS\_H**  
config.h, [566](#)

**\_\_FFLASFFPACK\_HAVE\_STRING\_H**  
config.h, [566](#)

**\_\_FFLASFFPACK\_HAVE\_SYS\_STAT\_H**  
config.h, [566](#)

- \_\_FFLASFFPACK\_HAVE\_SYS\_TIME\_H
  - config.h, [566](#)
- \_\_FFLASFFPACK\_HAVE\_SYS\_TYPES\_H
  - config.h, [566](#)
- \_\_FFLASFFPACK\_HAVE\_UNISTD\_H
  - config.h, [566](#)
- \_\_FFLASFFPACK\_KAAPI\_ROUTINES\_INL
  - kaapi\_routines.inl, [776](#)
- \_\_FFLASFFPACK\_LT\_OBJDIR
  - config.h, [566](#)
- \_\_FFLASFFPACK\_MINBLOCKCUTS
  - blockcuts.inl, [775](#)
- \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET
  - benchmark-charpoly.C, [524](#)
  - benchmark-fadd-lvl2.C, [529](#)
  - benchmark-fdot.C, [530](#)
  - benchmark-fgemm-mp.C, [531](#)
  - benchmark-fgemm-rns.C, [532](#)
  - benchmark-fgemv-mp.C, [534](#)
  - benchmark-fgemv.C, [535](#)
  - benchmark-fgesv.C, [538](#)
  - benchmark-fsyrc.C, [539](#)
  - benchmark-fsytrf.C, [540](#)
  - benchmark-ftrsm-mp.C, [540](#)
  - benchmark-ftrsm.C, [541](#)
  - benchmark-ftrsv.C, [542](#)
  - benchmark-ftrtri.C, [542](#)
  - benchmark-pluq.C, [545](#)
  - benchmark-quasisep.C, [546](#)
- \_\_FFLASFFPACK\_OPENBLAS\_NUM\_THREADS
  - config.h, [566](#)
- \_\_FFLASFFPACK\_PACKAGE
  - config.h, [566](#)
- \_\_FFLASFFPACK\_PACKAGE\_BUGREPORT
  - config.h, [566](#)
- \_\_FFLASFFPACK\_PACKAGE\_NAME
  - config.h, [566](#)
- \_\_FFLASFFPACK\_PACKAGE\_STRING
  - config.h, [566](#)
- \_\_FFLASFFPACK\_PACKAGE\_TARNAME
  - config.h, [567](#)
- \_\_FFLASFFPACK\_PACKAGE\_URL
  - config.h, [567](#)
- \_\_FFLASFFPACK\_PACKAGE\_VERSION
  - config.h, [567](#)
- \_\_FFLASFFPACK\_PLUQ\_THRESHOLD
  - fflas-ffpack-default-thresholds.h, [568](#)
  - test-echelon.C, [803](#)
- \_\_FFLASFFPACK\_SEQPARTHRESHOLD
  - fflas\_pfgemm.inl, [614](#)
- \_\_FFLASFFPACK\_SEQUENTIAL
  - parallel.h, [777](#)
  - test-echelon.C, [803](#)
  - test-ftrmm.C, [823](#)
  - test-ftrmv.C, [824](#)
  - test-ftrsm.C, [826](#)
  - test-ftrsv.C, [829](#)
  - test-ftrtri.C, [830](#)
  - test-lu.C, [833](#)
  - test-rankprofiles.C, [843](#)
- \_\_FFLASFFPACK\_SIZEOF\_CHAR
  - config.h, [567](#)
- \_\_FFLASFFPACK\_SIZEOF\_INT
  - config.h, [567](#)
- \_\_FFLASFFPACK\_SIZEOF\_LONG
  - config.h, [567](#)
- \_\_FFLASFFPACK\_SIZEOF\_LONG\_LONG
  - config.h, [567](#)
- \_\_FFLASFFPACK\_SIZEOF\_SHORT
  - config.h, [567](#)
- \_\_FFLASFFPACK\_SIZEOF\_\_\_INT64\_T
  - config.h, [567](#)
- \_\_FFLASFFPACK\_STDC\_HEADERS
  - config.h, [567](#)
- \_\_FFLASFFPACK\_USE\_OPENMP
  - config.h, [567](#)
- \_\_FFLASFFPACK\_VERSION
  - config.h, [567](#)
- \_\_FFLASFFPACK\_WINOTHRESHOLD
  - fflas-ffpack-default-thresholds.h, [568](#)
- \_\_FFLASFFPACK\_WINOTHRESHOLD\_BAL
  - fflas-ffpack-default-thresholds.h, [568](#)
- \_\_FFLASFFPACK\_WINOTHRESHOLD\_BAL\_FLT
  - fflas-ffpack-default-thresholds.h, [568](#)
- \_\_FFLASFFPACK\_WINOTHRESHOLD\_FLT
  - fflas-ffpack-default-thresholds.h, [568](#)
- \_\_FFLASFFPACK\_charpoly\_INL
  - ffpack\_charpoly.inl, [671](#)
- \_\_FFLASFFPACK\_checker\_charpoly\_INL
  - checker\_charpoly.inl, [550](#)
- \_\_FFLASFFPACK\_checker\_det\_INL
  - checker\_det.inl, [550](#)
- \_\_FFLASFFPACK\_checker\_fgemm\_INL
  - checker\_fgemm.inl, [551](#)
- \_\_FFLASFFPACK\_checker\_ftrsm\_INL
  - checker\_ftrsm.inl, [551](#)
- \_\_FFLASFFPACK\_checker\_invert\_INL
  - checker\_invert.inl, [551](#)
- \_\_FFLASFFPACK\_checker\_pluq\_INL
  - checker\_pluq.inl, [552](#)
- \_\_FFLASFFPACK\_fadd\_INL
  - fflas\_fadd.inl, [574](#)
- \_\_FFLASFFPACK\_fassign\_INL
  - fflas\_fassign.inl, [575](#)
- \_\_FFLASFFPACK\_faxpy\_INL
  - fflas\_faxpy.inl, [576](#)
- \_\_FFLASFFPACK\_fdot\_INL
  - fflas\_fdot.inl, [577](#)
- \_\_FFLASFFPACK\_fflas\_blockcuts\_INL
  - blockcuts.inl, [775](#)
- \_\_FFLASFFPACK\_fflas\_bounds\_INL
  - fflas\_bounds.inl, [571](#)
- \_\_FFLASFFPACK\_fflas\_fflas\_fgemm\_winograd\_INL
  - fgemm\_winograd.inl, [583](#)
- \_\_FFLASFFPACK\_fflas\_fflas\_level1\_INL
  - fflas\_level1.inl, [608](#)

- \_\_FFLASFFPACK\_fflas\_fflas\_level2\_INL  
fflas\_level2.inl, [611](#)
- \_\_FFLASFFPACK\_fflas\_fflas\_level3\_INL  
fflas\_level3.inl, [614](#)
- \_\_FFLASFFPACK\_fflas\_fflas\_mmhelper\_INL  
fflas\_helpers.inl, [603](#)
- \_\_FFLASFFPACK\_fflas\_fflas\_sparse\_INL  
fflas\_sparse.inl, [631](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_INL  
simd128.inl, [617](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_double\_INL  
simd128\_double.inl, [618](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_float\_INL  
simd128\_float.inl, [618](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_int16\_INL  
simd128\_int16.inl, [619](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_int32\_INL  
simd128\_int32.inl, [619](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_int64\_INL  
simd128\_int64.inl, [620](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_INL  
simd256.inl, [620](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_double\_INL  
simd256\_double.inl, [621](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_float\_INL  
simd256\_float.inl, [621](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_int16\_INL  
simd256\_int16.inl, [622](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_int32\_INL  
simd256\_int32.inl, [622](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_int64\_INL  
simd256\_int64.inl, [623](#)
- \_\_FFLASFFPACK\_fflas\_freduce\_INL  
fflas\_freduce.inl, [593](#)
- \_\_FFLASFFPACK\_fflas\_freduce\_mp\_INL  
fflas\_freduce\_mp.inl, [593](#)
- \_\_FFLASFFPACK\_fflas\_fsyr2k\_INL  
fflas\_fsyr2k.inl, [596](#)
- \_\_FFLASFFPACK\_fflas\_fsyrrk\_INL  
fflas\_fsyrrk.inl, [598](#)
- \_\_FFLASFFPACK\_fflas\_fsyrrk\_strassen\_INL  
fflas\_fsyrrk\_strassen.inl, [599](#)
- \_\_FFLASFFPACK\_fflas\_igemm\_igemm\_INL  
igemm.inl, [604](#)
- \_\_FFLASFFPACK\_fflas\_igemm\_igemm\_kernels\_INL  
igemm\_kernels.inl, [605](#)
- \_\_FFLASFFPACK\_fflas\_igemm\_igemm\_tools\_INL  
igemm\_tools.inl, [606](#)
- \_\_FFLASFFPACK\_fflas\_pfgemm\_INL  
fflas\_pfgemm.inl, [614](#)
- \_\_FFLASFFPACK\_fflas\_pftsrsm\_INL  
fflas\_pftsrsm.inl, [615](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_pspmm\_INL  
csr\_hyb\_pspmm.inl, [640](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_pspmv\_INL  
csr\_hyb\_pspmv.inl, [640](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_spm INL  
csr\_hyb\_spm.inl, [641](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_spmv\_INL  
csr\_hyb\_spmv.inl, [641](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_utils\_INL  
csr\_hyb\_utils.inl, [642](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_pspmm\_INL  
csr\_pspmm.inl, [636](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_pspmv\_INL  
csr\_pspmv.inl, [636](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_spm INL  
csr\_spm.inl, [637](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_spmv\_INL  
csr\_spmv.inl, [638](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_pspmm\_INL  
ell\_pspmm.inl, [643](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_pspmv\_INL  
ell\_pspmv.inl, [644](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_simd\_pspmv\_INL  
ell\_simd\_pspmv.inl, [648](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_simd\_spmv\_INL  
ell\_simd\_spmv.inl, [648](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_simd\_utils\_INL  
ell\_simd\_utils.inl, [649](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_spm INL  
ell\_spm.inl, [645](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_spmv\_INL  
ell\_spmv.inl, [646](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_utils\_INL  
ell\_utils.inl, [646](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_pspmm\_INL  
hyb\_zo\_pspmm.inl, [650](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_pspmv\_INL  
hyb\_zo\_pspmv.inl, [650](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_spm INL  
hyb\_zo\_spm.inl, [651](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_spmv\_INL  
hyb\_zo\_spmv.inl, [651](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_utils\_INL  
hyb\_zo\_utils.inl, [652](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_coo\_spm INL  
coo\_spm.inl, [633](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_coo\_spmv\_INL  
coo\_spmv.inl, [634](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_coo\_utils\_INL  
coo\_utils.inl, [634](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_sell\_pspmv\_INL  
sell\_pspmv.inl, [654](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_sell\_spmv\_INL  
sell\_spmv.inl, [655](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_sell\_utils\_INL  
sell\_utils.inl, [655](#)
- \_\_FFLASFFPACK\_ffpack\_INL  
ffpack.inl, [669](#)
- \_\_FFLASFFPACK\_ffpack\_bruhatgen\_inl  
ffpack\_bruhatgen.inl, [670](#)
- \_\_FFLASFFPACK\_ffpack\_charpoly\_danilveski\_INL  
ffpack\_charpoly\_danilevski.inl, [671](#)
- \_\_FFLASFFPACK\_ffpack\_charpoly\_kgfast\_INL  
ffpack\_charpoly\_kgfast.inl, [672](#)

\_\_FFLASFFPACK\_ffpack\_charpoly\_kgfastgeneralized\_INL  
     ffpack\_charpoly\_kgfastgeneralized.inl, 672  
 \_\_FFLASFFPACK\_ffpack\_charpoly\_kglu\_INL  
     ffpack\_charpoly\_kglu.inl, 673  
 \_\_FFLASFFPACK\_ffpack\_echelon\_forms\_INL  
     ffpack\_echelonforms.inl, 675  
 \_\_FFLASFFPACK\_ffpack\_fgesv\_INL  
     ffpack\_fgesv.inl, 676  
 \_\_FFLASFFPACK\_ffpack\_fgetrs\_INL  
     ffpack\_fgetrs.inl, 676  
 \_\_FFLASFFPACK\_ffpack\_fsytrf\_INL  
     ffpack\_fsytrf.inl, 678  
 \_\_FFLASFFPACK\_ffpack\_ftrssyr2k\_INL  
     ffpack\_ftrssyr2k.inl, 679  
 \_\_FFLASFFPACK\_ffpack\_ftrstr\_INL  
     ffpack\_ftrstr.inl, 679  
 \_\_FFLASFFPACK\_ffpack\_ftrtr\_INL  
     ffpack\_ftrtr.inl, 680  
 \_\_FFLASFFPACK\_ffpack\_invert\_INL  
     ffpack\_invert.inl, 680  
 \_\_FFLASFFPACK\_ffpack\_krylovelim\_INL  
     ffpack\_krylovelim.inl, 681  
 \_\_FFLASFFPACK\_ffpack\_ludivine\_INL  
     ffpack\_ludivine.inl, 681  
 \_\_FFLASFFPACK\_ffpack\_minpoly\_INL  
     ffpack\_minpoly.inl, 683  
 \_\_FFLASFFPACK\_ffpack\_permutation\_INL  
     ffpack\_permutation.inl, 685  
 \_\_FFLASFFPACK\_ffpack\_pluq\_INL  
     ffpack\_pluq.inl, 686  
 \_\_FFLASFFPACK\_ffpack\_ppluq\_INL  
     ffpack\_ppluq.inl, 687  
 \_\_FFLASFFPACK\_ffpack\_rank\_profiles\_INL  
     ffpack\_rankprofiles.inl, 688  
 \_\_FFLASFFPACK\_ffgemm\_INL  
     fflas\_fgemm.inl, 579  
 \_\_FFLASFFPACK\_ffgemm\_bini\_INL  
     schedule\_bini.inl, 583  
 \_\_FFLASFFPACK\_ffgemm\_winograd\_INL  
     schedule\_winograd.inl, 584  
 \_\_FFLASFFPACK\_ffgemm\_winograd\_acc\_INL  
     schedule\_winograd\_acc.inl, 585  
 \_\_FFLASFFPACK\_ffgemm\_winograd\_acc\_ip\_INL  
     schedule\_winograd\_acc\_ip.inl, 585  
 \_\_FFLASFFPACK\_ffgemm\_winograd\_ip\_INL  
     schedule\_winograd\_ip.inl, 586  
 \_\_FFLASFFPACK\_ffgemv\_INL  
     fflas\_ffgemv.inl, 587  
 \_\_FFLASFFPACK\_ffgemv\_mp\_INL  
     fflas\_ffgemv\_mp.inl, 588  
 \_\_FFLASFFPACK\_fger\_INL  
     fflas\_fger.inl, 589  
 \_\_FFLASFFPACK\_field\_rns\_INL  
     rns.inl, 695  
 \_\_FFLASFFPACK\_field\_rns\_double\_INL  
     rns-double.inl, 693  
 \_\_FFLASFFPACK\_field\_rns\_double\_recint\_INL  
     rns-double-recint.inl, 692  
 \_\_FFLASFFPACK\_freivalds\_INL  
     fflas\_freivalds.inl, 593  
 \_\_FFLASFFPACK\_fscal\_INL  
     fflas\_fscal.inl, 595  
 \_\_FFLASFFPACK\_fscal\_mp\_INL  
     fflas\_fscal\_mp.inl, 596  
 \_\_FFLASFFPACK\_ftrmm\_INL  
     fflas\_ftrmm.inl, 600  
 \_\_FFLASFFPACK\_ftrsm\_INL  
     fflas\_ftrsm.inl, 600  
 \_\_FFLASFFPACK\_ftrsv\_INL  
     fflas\_ftrsv.inl, 602  
 \_\_FFLASFFPACK\_simd512\_INL  
     simd512.inl, 623  
 \_\_FFLASFFPACK\_simd512\_double\_INL  
     simd512\_double.inl, 624  
 \_\_FFLASFFPACK\_simd512\_float\_INL  
     simd512\_float.inl, 624  
 \_\_FFLASFFPACK\_simd512\_int32\_INL  
     simd512\_int32.inl, 624  
 \_\_FFLAS\_L1\_INST\_C  
     fflas\_L1\_inst.C, 707  
 \_\_FFLAS\_L2\_INST\_C  
     fflas\_L2\_inst.C, 711  
 \_\_FFLAS\_L3\_INST\_C  
     fflas\_L3\_inst.C, 714  
 \_\_FFLAS\_\_TRSM\_READONLY  
     fflas\_L3\_inst\_implement.inl, 717  
     fflas\_level3.inl, 614  
     ffpack\_ppluq.inl, 687  
 \_\_FFPACK\_FSYTRF\_BC\_CROUT  
     benchmark-fsytrf.C, 540  
 \_\_FFPACK\_INST\_C  
     ffpack\_inst.C, 769  
 \_\_FFPACK\_charpoly\_mp\_INL  
     ffpack\_charpoly\_mp.inl, 673  
 \_\_FFPACK\_det\_mp\_INL  
     ffpack\_det\_mp.inl, 674  
 \_\_FFPACK\_ffgemm\_classical\_INL  
     ffgemm\_classical\_mp.inl, 581  
 \_\_FFPACK\_fger\_mp\_INL  
     fflas\_fger\_mp.inl, 590  
 \_\_FFPACK\_ftrsm\_mp\_INL  
     fflas\_ftrsm\_mp.inl, 601  
 \_\_FFPACK\_ludivine\_mp\_INL  
     ffpack\_ludivine\_mp.inl, 682  
 \_\_FFPACK\_pluq\_mp\_INL  
     ffpack\_pluq\_mp.inl, 686  
 \_\_LUDIVINE\_CUTOFF  
     test-lu.C, 834  
 \_\_has\_builtin  
     bit\_manipulation.h, 787  
 \_\_alloc  
     rns\_double\_elt, 469  
     rns\_double\_elt\_cstptr, 472  
     rns\_double\_elt\_ptr, 475  
 \_\_basis  
     rns\_double, 467

- rns\_double\_extended, 478
- \_basisMax
  - rns\_double, 467
  - rns\_double\_extended, 478
- \_coo
  - SpMat< Field, flag >, 501
- \_coo16
  - CooMat< Field >, 419
- \_coo16\_zo
  - CooMat< Field >, 419
- \_coo32
  - CooMat< Field >, 419
- \_coo32\_zo
  - CooMat< Field >, 419
- \_coo64
  - CooMat< Field >, 419
- \_coo64\_zo
  - CooMat< Field >, 419
- \_crt\_in
  - rns\_double, 468
  - rns\_double\_extended, 478
- \_crt\_out
  - rns\_double, 468
  - rns\_double\_extended, 478
- \_csr
  - SpMat< Field, flag >, 501
- \_csr16
  - CsrMat< Field >, 420
- \_csr16\_zo
  - CsrMat< Field >, 420
- \_csr32
  - CsrMat< Field >, 420
- \_csr32\_zo
  - CsrMat< Field >, 420
- \_csr64
  - CsrMat< Field >, 420
- \_csr64\_zo
  - CsrMat< Field >, 420
- \_ell
  - SpMat< Field, flag >, 501
- \_ell16
  - EllMat< Field >, 422
- \_ell16\_zo
  - EllMat< Field >, 422
- \_ell32
  - EllMat< Field >, 422
- \_ell32\_zo
  - EllMat< Field >, 422
- \_ell64
  - EllMat< Field >, 422
- \_ell64\_zo
  - EllMat< Field >, 422
- \_errorStream
  - Failure, 424
- \_field\_rns
  - rns\_double, 467
  - rns\_double\_extended, 478
- \_iM\_modp\_rns
  - RNSIntegerMod< RNS >, 488
- \_ibeg
  - ForStrategy2D< blocksize\_t, Cut, Param >, 436
- \_iend
  - ForStrategy2D< blocksize\_t, Cut, Param >, 436
- \_invbasis
  - rns\_double, 467
  - rns\_double\_extended, 478
- \_jbeg
  - ForStrategy2D< blocksize\_t, Cut, Param >, 437
- \_jend
  - ForStrategy2D< blocksize\_t, Cut, Param >, 437
- \_ldm
  - rns\_double, 468
  - rns\_double\_extended, 478
- \_mi\_sum
  - rns\_double, 468
- \_mm
  - Test< Elt >, 508
- \_negbasis
  - rns\_double, 467
  - rns\_double\_extended, 478
- \_nn
  - Test< Elt >, 508
- \_p
  - RNSIntegerMod< RNS >, 488
- \_pbits
  - rns\_double, 468
  - rns\_double\_extended, 478
- \_ptr
  - rns\_double\_elt, 469
  - rns\_double\_elt\_cstptr, 472
  - rns\_double\_elt\_ptr, 475
- \_rns
  - RNSInteger< RNS >, 482
  - RNSIntegerMod< RNS >, 488
- \_simd512\_int64\_INL
  - simd512\_int64.inl, 625
- \_size
  - rns\_double, 468
  - rns\_double\_extended, 478
- \_stride
  - rns\_double\_elt, 469
  - rns\_double\_elt\_cstptr, 472
  - rns\_double\_elt\_ptr, 475
- ~CheckerImplem\_Det
  - CheckerImplem\_Det< Field >, 406
- ~CheckerImplem\_PLUQ
  - CheckerImplem\_PLUQ< Field >, 411
- ~CheckerImplem\_charpoly
  - CheckerImplem\_charpoly< Field, Polynomial >, 406
- ~CheckerImplem\_fgemm
  - CheckerImplem\_fgemm< Field >, 408
- ~CheckerImplem\_ftsm
  - CheckerImplem\_ftsm< Field >, 409
- ~CheckerImplem\_invert
  - CheckerImplem\_invert< Field >, 410

- ~rns\_double\_elt
  - rns\_double\_elt, [469](#)
- 101-fgemv.C, [849](#)
  - main, [849](#)
- 2x2-fgemv.C, [850](#)
  - main, [850](#)
- 2x2-ftsrv.C, [850](#)
  - main, [850](#)
- 2x2-pluq.C, [850](#)
  - main, [851](#)
- add
  - FFLAS::vectorised, [280](#)
  - FieldSimd< \_Field >, [427](#)
  - RNSIntegerMod< RNS >, [486](#)
  - ScalFunctions< Element >, [492](#)
- add\_r
  - FieldSimd< \_Field >, [427](#)
- addin
  - FieldSimd< \_Field >, [427](#)
  - ScalFunctions< Element >, [492](#)
- addin\_r
  - FieldSimd< \_Field >, [427](#)
- addp
  - FFLAS::vectorised, [279](#)
- AlgoChooser< ModeT, ParSeq >, [397](#)
  - value, [397](#)
- align-allocator.h, [784](#)
- alignable
  - FFLAS, [188](#)
- alignable< Givaro::Integer \* >
  - FFLAS, [188](#)
- aligned\_allocator
  - NoSimd< T >, [458](#)
- aligned\_vector
  - NoSimd< T >, [458](#)
- alignment
  - FieldSimd< \_Field >, [430](#)
  - NoSimd< T >, [459](#)
- ALL< v >, [397](#)
- Amax
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [455](#)
- Amin
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [455](#)
- applyP
  - FFPACK, [304](#), [305](#), [363](#)
- applyP\_block
  - FFPACK, [355](#)
- applyP\_modular\_double
  - ffpack.C, [734](#)
  - ffpack\_c.h, [755](#)
- ArbitraryPreIntTag, [397](#)
- Architecture of the library., [2](#)
- areEqual
  - RNSIntegerMod< RNS >, [487](#)
- AreEqual< X, Y >, [397](#)
  - value, [398](#)
- args-parser.h, [784](#)
  - ArgumentType, [785](#)
  - END\_OF\_ARGUMENTS, [785](#)
  - findArgument, [786](#)
  - getListArgs, [786](#)
  - printHelpMessage, [786](#)
  - TYPE\_BOOL, [785](#)
  - TYPE\_DOUBLE, [786](#)
  - TYPE\_INT, [786](#)
  - TYPE\_INTEGER, [786](#)
  - type\_integer, [785](#)
  - TYPE\_INTLIST, [786](#)
  - TYPE\_LONGLONG, [786](#)
  - TYPE\_NONE, [785](#)
  - TYPE\_STR, [786](#)
  - TYPE\_UINT64, [786](#)
- Argument, [398](#)
  - c, [398](#)
  - data, [398](#)
  - example, [398](#)
  - helpString, [398](#)
  - type, [398](#)
- ArgumentType
  - args-parser.h, [785](#)
- ArithProg
  - FFPACK::Protected, [393](#)
- arithprog.C, [517](#)
  - main, [517](#)
  - TTimer, [517](#)
- array< T >, [398](#)
  - elements, [399](#)
- array< T >::const\_iterator, [413](#)
- array< T >::const\_reverse\_iterator, [413](#)
- array< T >::iterator, [450](#)
- array< T >::reverse\_iterator, [463](#)
- assign
  - RNSInteger< RNS >, [482](#)
  - RNSIntegerMod< RNS >, [486](#)
- associatedDelayedField< Field >, [399](#)
  - field, [399](#)
  - type, [399](#)
- assume\_aligned
  - fflas\_sparse.h, [629](#)
- AtlasConj
  - config-blas.h, [556](#)
- Aunfit
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [454](#)
- aut
  - HelperFlag, [445](#)
- Auto, [399](#)
- averageCol
  - StatsMatrix, [502](#)
- averageColDifference
  - StatsMatrix, [503](#)
- averageRow
  - StatsMatrix, [502](#)
- averageRowDifference



- StatsMatrix, [503](#)
- axpy
  - FieldSimd< \_Field >, [429](#)
- axpy\_r
  - FieldSimd< \_Field >, [430](#)
- axpyin
  - FieldSimd< \_Field >, [429](#)
  - RNSIntegerMod< RNS >, [487](#)
- axpyin\_r
  - FieldSimd< \_Field >, [430](#)
- axpyp
  - FFLAS::vectorised, [280](#)
  - FFLAS::vectorised::unswitch, [284](#)
- balanced
  - FieldTraits< Field >, [431](#)
  - winograd.C, [523](#)
- BARRIER
  - parallel.h, [777](#)
- BASECASE\_K
  - test-lu.C, [833](#)
- BaseTimer
  - FFLAS, [74](#)
- BasisElement
  - rns\_double, [465](#)
  - rns\_double\_extended, [476](#)
  - RNSInteger< RNS >, [480](#)
  - RNSIntegerMod< RNS >, [484](#)
- begin
  - ForStrategy1D< blocksize\_t, Cut, Param >, [433](#)
  - Info, [447](#), [448](#)
- BEGIN\_PARALLEL\_MAIN
  - parallel.h, [778](#)
- Bench
  - Bench< Elt >, [401](#)
- Bench< Elt >, [399](#)
  - Bench, [401](#)
  - cardinality, [401](#)
  - doBenchs, [401](#)
  - Elt\_ptr, [400](#)
  - enable\_if\_no\_simd\_t, [400](#)
  - enable\_if\_simd128\_t, [401](#)
  - enable\_if\_simd256\_t, [401](#)
  - enable\_if\_simd512\_t, [401](#)
  - enable\_if\_t, [400](#)
  - F, [402](#)
  - Field, [400](#)
  - inplace, [402](#)
  - is\_same\_element, [400](#)
  - iters, [402](#)
  - m, [402](#)
  - n, [402](#)
  - Residu, [400](#)
  - run, [401](#)
- benchmark-charpoly-mp.C, [523](#)
  - \_\_FFLASFFPACK\_FORCE\_SEQ, [523](#)
  - main, [523](#)
- benchmark-charpoly.C, [524](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [524](#)
  - main, [524](#)
  - run\_with\_field, [524](#)
- benchmark-checkers.C, [524](#)
  - \_\_MAX\_SIZE\_MATRICES, [525](#)
  - \_\_NR\_TESTS, [525](#)
  - CUBE, [525](#)
  - ENABLE\_ALL\_CHECKINGS, [525](#)
  - main, [525](#)
- benchmark-dgemm.C, [525](#)
  - CBLAS\_GEMM, [526](#)
  - Floats, [526](#)
  - main, [526](#)
  - TTimer, [526](#)
- benchmark-dgetrf.C, [526](#)
  - \_\_FFLASFFPACK\_HAVE\_DGETRF, [527](#)
  - main, [527](#)
- benchmark-dgetri.C, [527](#)
  - main, [527](#)
- benchmark-dsytrf.C, [527](#)
  - EFFGFF, [528](#)
  - main, [528](#)
- benchmark-dtrsm.C, [528](#)
  - main, [528](#)
- benchmark-dtrtri.C, [528](#)
  - \_\_FFLASFFPACK\_HAVE\_DTRTRI, [529](#)
  - main, [529](#)
- benchmark-fadd-lvl2.C, [529](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [529](#)
  - main, [529](#)
- benchmark-fdot.C, [529](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [530](#)
  - main, [530](#)
  - run\_with\_field, [530](#)
- benchmark-fgemm-mp.C, [530](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [531](#)
  - main, [531](#)
  - MG\_DEFAULT, [531](#)
  - STD\_RECINT\_SIZE, [531](#)
  - tmain, [531](#)
- benchmark-fgemm-rns.C, [531](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [532](#)
  - ConstElement\_ptr, [532](#)
  - Element\_ptr, [532](#)
  - Field, [532](#)
  - GRAIN, [532](#)
  - main, [533](#)
  - PSeq, [533](#)
  - RNS, [532](#)
  - THREADS, [532](#)
  - THREED, [532](#)
  - THREEDA, [532](#)
  - THREEDIP, [532](#)

- TWOD, [532](#)
- TWODA, [532](#)
- benchmark-fgemm.C, [533](#)
  - CLASSIC\_HYBRID, [533](#)
  - main, [533](#)
- benchmark-fgemv-mp.C, [533](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [534](#)
  - main, [534](#)
  - MG\_DEFAULT, [534](#)
  - STD\_RECINT\_SIZE, [534](#)
  - tmain, [534](#)
  - write\_matrix, [534](#)
- benchmark-fgemv.C, [535](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [535](#)
  - benchmark\_disp, [537](#)
  - benchmark\_in\_Field, [537](#)
  - benchmark\_with\_field, [537](#)
  - benchmark\_with\_timer, [536](#)
  - check\_result, [536](#)
  - fill\_value, [536](#)
  - genData, [536](#)
  - main, [538](#)
- benchmark-fgesv.C, [538](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [538](#)
  - main, [538](#)
- benchmark-fsyr2k.C, [538](#)
  - main, [539](#)
- benchmark-fsyrk.C, [539](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [539](#)
  - main, [539](#)
- benchmark-fsytrf.C, [539](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [540](#)
  - \_\_FFPACK\_FSYTRF\_BC\_CROUT, [540](#)
  - CUBE, [540](#)
  - main, [540](#)
- benchmark-ftdsm-mp.C, [540](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [540](#)
  - main, [541](#)
- benchmark-ftdsm.C, [541](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [541](#)
  - main, [541](#)
- benchmark-ftsv.C, [541](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [542](#)
  - main, [542](#)
- benchmark-fttri.C, [542](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [542](#)
  - CUBE, [542](#)
  - main, [542](#)
- benchmark-inverse.C, [542](#)
- CUBE, [543](#)
  - main, [543](#)
- benchmark-lqup-mp.C, [543](#)
  - main, [543](#)
- benchmark-lqup.C, [544](#)
  - CUBE, [544](#)
  - main, [544](#)
- benchmark-pluq.C, [544](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [545](#)
  - CUBE, [545](#)
  - Field, [545](#)
  - main, [545](#)
  - Rec\_Initialize, [545](#)
  - verification\_PLUQ, [545](#)
- benchmark-quasisep.C, [545](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [546](#)
  - main, [546](#)
  - run\_with\_field, [546](#)
- benchmark-storage-transpose.C, [546](#)
  - main, [547](#)
- benchmark-wino.C, [547](#)
  - CUBE, [547](#)
  - launch\_wino, [547](#)
  - main, [548](#)
- benchmark\_disp
- benchmark-fgemv.C, [537](#)
- benchmark\_in\_Field
- benchmark-fgemv.C, [537](#)
- benchmark\_with\_field
- benchmark-fgemv.C, [537](#)
- benchmark\_with\_timer
- benchmark-fgemv.C, [536](#)
- Bibliography, [7](#)
- Bini, [402](#)
- FFLAS::BLAS3, [192](#)
- bit\_manipulation.h, [786](#)
  - \_\_has\_builtin, [787](#)
  - clz, [787](#)
  - ctz, [787](#)
- bitsize
- FFLAS, [140](#)
- bitsize< Givaro::ZRing< Givaro::Integer > >
- FFLAS, [140](#)
- config-blas.h, [555](#)
- blend
- ScalFunctions< Element >, [495](#)
- Block, [402](#)
- 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, [774](#)
  - \_\_FFLASFFPACK\_MINBLOCKCUTS, [775](#)
  - \_\_FFLASFFPACK\_fflas\_blockcuts\_INL, [775](#)
- blockindex
  - ForStrategy1D< blocksize\_t, Cut, Param >, [433](#)
  - ForStrategy2D< blocksize\_t, Cut, Param >, [436](#)
- BlockingFactor
  - FFLAS::details, [206](#)
- BLOCKS
  - ForStrategy2D< blocksize\_t, Cut, Param >, [438](#)
- BlockTransposeSIMD< Field, Simd, >, [402](#)
  - info, [403](#)
  - size, [403](#)
  - transpose, [403](#), [404](#)
- Bmax
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, [455](#)
- Bmin
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, [455](#)
- Bruhat2EchelonPermutation
  - FFPACK, [343](#)
- Bug List, [3](#)
- build
  - ForStrategy1D< blocksize\_t, Cut, Param >, [432](#)
- buildMatrix
  - FFPACK, [349](#)
- Bunfit
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, [454](#)
- c
  - Argument, [398](#)
- callLUdivine\_small< Element >, [404](#)
  - operator(), [404](#)
- cardinality
  - Bench< Elt >, [401](#)
  - RNSInteger< RNS >, [481](#)
  - RNSIntegerMod< RNS >, [485](#)
  - Test< Elt >, [507](#)
- cast.h, [787](#)
- category
  - FieldTraits< Field >, [431](#)
- cblas.C, [794](#)
  - \_\_FFLASFFPACK\_CONFIGURATION, [795](#)
  - \_\_FFLASFFPACK\_HAVE\_CBLAS, [795](#)
  - main, [795](#)
- CBLAS\_DIAG
  - config-blas.h, [556](#)
- cblas\_dsyrc
  - config-blas.h, [560](#)
- CBLAS\_ENUM\_DEFINED\_H
  - config-blas.h, [555](#)
- CBLAS\_EXTERNALS
  - config-blas.h, [555](#)
- CBLAS\_GEMM
  - benchmark-dgemm.C, [526](#)
- cblas\_imprsm
  - FFLAS, [127](#)
- CBLAS\_INT
  - config-blas.h, [555](#)
- CBLAS\_ORDER
  - config-blas.h, [555](#)
- CBLAS\_SIDE
  - config-blas.h, [556](#)
- CBLAS\_TRANSPOSE
  - config-blas.h, [555](#)
- CBLAS\_UPLO
  - config-blas.h, [556](#)
- CblasColMajor
  - config-blas.h, [555](#)
- CblasConjTrans
  - config-blas.h, [556](#)
- CblasLeft
  - config-blas.h, [556](#)
- CblasLower
  - config-blas.h, [556](#)
- CblasNonUnit
  - config-blas.h, [556](#)
- CblasNoTrans
  - config-blas.h, [556](#)
- CblasRight
  - config-blas.h, [556](#)
- CblasRowMajor
  - config-blas.h, [555](#)
- CblasTrans
  - config-blas.h, [556](#)
- CblasUnit
  - config-blas.h, [556](#)
- CblasUpper
  - config-blas.h, [556](#)
- changeBS
  - ForStrategy1D< blocksize\_t, Cut, Param >, [434](#)

- changeCBS
  - ForStrategy2D< blocksize\_t, Cut, Param >, 437
- changeRBS
  - ForStrategy2D< blocksize\_t, Cut, Param >, 437
- characteristic
  - RNSInteger< RNS >, 481
  - RNSIntegerMod< RNS >, 485
- CharPoly
  - FFPACK, 321, 322, 349, 367, 368
- charpoly.C, 517, 518
  - CUBE, 518
  - GFOPS, 518
  - main, 518
  - TTimer, 518
- CharpolyFailed, 404
- check
  - Checker\_Empty< Field >, 405
  - CheckerImplem\_charpoly< Field, Polynomial >, 406
  - CheckerImplem\_Det< Field >, 407
  - CheckerImplem\_fgemm< Field >, 408
  - CheckerImplem\_ftsm< Field >, 409
  - CheckerImplem\_invert< Field >, 410
  - CheckerImplem\_PLUQ< Field >, 411
- check1
  - regression-check.C, 797
- check2
  - regression-check.C, 797
- check3
  - regression-check.C, 798
- check4
  - regression-check.C, 798
- check\_computeS1S2
  - test-fsyk.C, 820
- CHECK\_DEPENDENCIES
  - parallel.h, 777
- check\_eq
  - test-simd.C, 847
- check\_fdot
  - test-fdot.C, 806
- check\_fger
  - test-fger.C, 813
- check\_fsyk2k
  - test-fsyk2k.C, 818
- check\_fsyk
  - test-fsyk.C, 820
- check\_fsyk\_bkdiag
  - test-fsyk.C, 820
- check\_fsyk\_diag
  - test-fsyk.C, 820
- check\_ftmm
  - test-ftmm.C, 823
- check\_ftmv
  - test-ftmv.C, 824
- check\_ftsm
  - test-ftsm.C, 826
- check\_ftssyr2k
  - test-ftssyr2k.C, 827
- check\_ftstr
  - test-ftstr.C, 828
- check\_ftsv
  - test-ftsv.C, 829
- check\_fttri
  - test-fttri.C, 830
- check\_minpoly
  - test-minpoly.C, 838
- check\_MM
  - test-fgemm.C, 809
- check\_MV
  - test-fgemv.C, 811
- check\_result
  - benchmark-fgemv.C, 536
- check\_solve
  - test-solve.C, 848
- checkA
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, 454
- checkB
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, 454
- CHECKER, 41
- Checker\_charpoly
  - FFPACK, 303
- checker\_charpoly.inl, 549
  - \_\_FFLASFFPACK\_checker\_charpoly\_INL, 550
- Checker\_Det
  - FFPACK, 302
- checker\_det.inl, 550
  - \_\_FFLASFFPACK\_checker\_det\_INL, 550
- Checker\_Empty
  - Checker\_Empty< Field >, 405
- Checker\_Empty< Field >, 405
  - check, 405
  - Checker\_Empty, 405
- checker\_empty.h, 550
- Checker\_fgemm
  - FFLAS, 72
- checker\_fgemm.inl, 550
  - \_\_FFLASFFPACK\_checker\_fgemm\_INL, 551
- Checker\_ftsm
  - FFLAS, 72
- checker\_ftsm.inl, 551
  - \_\_FFLASFFPACK\_checker\_ftsm\_INL, 551
- Checker\_invert
  - FFPACK, 303
- checker\_invert.inl, 551
  - \_\_FFLASFFPACK\_checker\_invert\_INL, 551
- Checker\_PLUQ
  - FFPACK, 302
- checker\_pluq.inl, 551
  - \_\_FFLASFFPACK\_checker\_pluq\_INL, 552
- CheckerImplem\_charpoly
  - CheckerImplem\_charpoly< Field, Polynomial >, 405
- CheckerImplem\_charpoly< Field, Polynomial >, 405
  - ~CheckerImplem\_charpoly, 406

- check, [406](#)
- CheckerImplem\_charpoly, [405](#)
- CheckerImplem\_Det
  - CheckerImplem\_Det< Field >, [406](#)
- CheckerImplem\_Det< Field >, [406](#)
- ~CheckerImplem\_Det, [406](#)
- check, [407](#)
- CheckerImplem\_Det, [406](#)
- CheckerImplem\_fgemm
  - CheckerImplem\_fgemm< Field >, [407](#)
- CheckerImplem\_fgemm< Field >, [407](#)
- ~CheckerImplem\_fgemm, [408](#)
- check, [408](#)
- CheckerImplem\_fgemm, [407](#)
- CheckerImplem\_ftrsm
  - CheckerImplem\_ftrsm< Field >, [408](#), [409](#)
- CheckerImplem\_ftrsm< Field >, [408](#)
- ~CheckerImplem\_ftrsm, [409](#)
- check, [409](#)
- CheckerImplem\_ftrsm, [408](#), [409](#)
- CheckerImplem\_invert
  - CheckerImplem\_invert< Field >, [409](#), [410](#)
- CheckerImplem\_invert< Field >, [409](#)
- ~CheckerImplem\_invert, [410](#)
- check, [410](#)
- CheckerImplem\_invert, [409](#), [410](#)
- CheckerImplem\_PLUQ
  - CheckerImplem\_PLUQ< Field >, [410](#)
- CheckerImplem\_PLUQ< Field >, [410](#)
- ~CheckerImplem\_PLUQ, [411](#)
- check, [411](#)
- CheckerImplem\_PLUQ, [410](#)
- checkers.doxy, [552](#)
- checkers\_fflas.h, [552](#)
- checkers\_fflas.inl, [552](#)
- FFLASFFPACK\_checkers\_fflas\_inl\_H, [553](#)
- checkers\_ffpack.h, [553](#)
- checkers\_ffpack.inl, [553](#)
- FFLASFFPACK\_checkers\_ffpack\_inl\_H, [554](#)
- checkingMessage
  - test-nullspace.C, [839](#)
- checkMonotonicApplyP
  - test-permutations.C, [840](#)
- checkOut
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [454](#)
- checkRPM
  - test-rpm.C, [844](#)
- checkSymmetricRPM
  - test-rpm.C, [844](#)
- checkZeroDimCharpoly
  - regression-check.C, [798](#)
- checkZeroDimMinPoly
  - regression-check.C, [798](#)
- chooseField
  - FFPACK, [388](#)
- chooseField< Givaro::ZRing< double > >
  - FFPACK, [389](#)
- chooseField< Givaro::ZRing< float > >
  - FFPACK, [388](#)
- chooseField< Givaro::ZRing< int32\_t > >
  - FFPACK, [388](#)
- chooseField< Givaro::ZRing< int64\_t > >
  - FFPACK, [388](#)
- clapack.C, [795](#)
  - \_\_FFLASFFPACK\_CONFIGURATION, [795](#)
  - \_\_FFLASFFPACK\_HAVE\_CLAPACK, [795](#)
  - \_\_FFLASFFPACK\_HAVE\_LAPACK, [795](#)
  - main, [795](#)
- Classic, [411](#)
- CLASSIC\_HYBRID
  - benchmark-fgemm.C, [533](#)
- clz
  - bit\_manipulation.h, [787](#)
- Cmax
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [455](#)
- Cmin
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [455](#)
- cmp
  - test-simd.C, [847](#)
- col
  - Coo< Field >, [417](#)
  - Coo< ValT, IdxT >, [415](#), [418](#)
- colblockindex
  - ForStrategy2D< blocksize\_t, Cut, Param >, [436](#)
- colBlockSize
  - ForStrategy2D< blocksize\_t, Cut, Param >, [437](#)
- coldim
  - StatsMatrix, [502](#)
- colnumblocks
  - ForStrategy2D< blocksize\_t, Cut, Param >, [436](#)
- ColRankProfileSubmatrix
  - FFPACK, [334](#), [372](#)
- ColRankProfileSubmatrix\_modular\_double
  - ffpack.C, [746](#)
  - ffpack\_c.h, [766](#)
- ColRankProfileSubmatrixIndices
  - FFPACK, [333](#), [372](#)
- ColRankProfileSubmatrixIndices\_modular\_double
  - ffpack.C, [746](#)
  - ffpack\_c.h, [765](#)
- Column, [411](#)
- ColumnEchelonForm
  - FFPACK, [315](#), [316](#), [366](#)
- ColumnEchelonForm\_modular\_double
  - ffpack.C, [736](#)
  - ffpack\_c.h, [758](#)
- ColumnEchelonForm\_modular\_float
  - ffpack.C, [737](#)
  - ffpack\_c.h, [759](#)
- ColumnEchelonForm\_modular\_int32\_t
  - ffpack.C, [738](#)
  - ffpack\_c.h, [759](#)
- ColumnRankProfile

- FFPACK, 330, 331, 371
- ColumnRankProfile\_modular\_double
  - ffpack.C, 745
  - ffpack\_c.h, 764
- COMMA
  - parallel.h, 780
- CompactElement< Element >, 411
  - type, 412
- compatible\_data\_type< Field >, 412
  - value, 412
- compliant
  - NoSimd< T >, 458
- Compose
  - Compose< H1, H2 >, 412, 413
- Compose< H1, H2 >, 412
  - Compose, 412, 413
  - first\_component, 413
  - operator<<, 413
  - second\_component, 413
- composePermutationsLLL
  - FFPACK, 358
  - ffpack.C, 733
  - ffpack\_c.h, 755
- composePermutationsLLM
  - FFPACK, 358
  - ffpack.C, 733
  - ffpack\_c.h, 754
- composePermutationsMLM
  - FFPACK, 359
  - ffpack.C, 733
  - ffpack\_c.h, 755
- CompressRows
  - FFPACK::Protected, 394
- CompressRowsQA
  - FFPACK::Protected, 395
- CompressRowsQK
  - FFPACK::Protected, 395
- CompressToBlockBiDiagonal
  - FFPACK, 342
- computeDeviation
  - FFLAS, 153
- computeFactorClassic
  - FFLAS::Protected, 211
- ComputeRPermutation
  - FFPACK, 344, 347
- computeS1S2
  - FFLAS, 123
- config-blas.h, 554
  - AtlasConj, 556
  - blas\_enum, 555
  - CBLAS\_DIAG, 556
  - cblas\_dsyrk, 560
  - CBLAS\_ENUM\_DEFINED\_H, 555
  - CBLAS\_EXTERNALS, 555
  - CBLAS\_INT, 555
  - CBLAS\_ORDER, 555
  - CBLAS\_SIDE, 556
  - CBLAS\_TRANSPOSE, 555
  - CBLAS\_UPLO, 556
  - CblasColMajor, 555
  - CblasConjTrans, 556
  - CblasLeft, 556
  - CblasLower, 556
  - CblasNonUnit, 556
  - CblasNoTrans, 556
  - CblasRight, 556
  - CblasRowMajor, 555
  - CblasTrans, 556
  - CblasUnit, 556
  - CblasUpper, 556
  - dasum\_, 557
  - daxpy\_, 556
  - dcopy\_, 558
  - ddot\_, 556
  - dgemm\_, 560
  - dgemv\_, 557
  - dger\_, 558
  - dnrm2\_, 557
  - dscal\_, 558
  - dtrmm\_, 559
  - dtrsm\_, 559
  - idamax\_, 557
  - saxpy\_, 556
  - scopy\_, 558
  - sdot\_, 557
  - sgemm\_, 560
  - sgemv\_, 557
  - sger\_, 558
  - sscal\_, 559
  - strmm\_, 559
  - strsm\_, 559
- config.h, 561, 564
  - \_\_FFLASFFPACK\_HAVE\_BLAS, 565
  - \_\_FFLASFFPACK\_HAVE\_CBLAS, 565
  - \_\_FFLASFFPACK\_HAVE\_CXX11, 565
  - \_\_FFLASFFPACK\_HAVE\_DLFCN\_H, 565
  - \_\_FFLASFFPACK\_HAVE\_FLOAT\_H, 565
  - \_\_FFLASFFPACK\_HAVE\_INT128, 565
  - \_\_FFLASFFPACK\_HAVE\_INTPTR\_T, 565
  - \_\_FFLASFFPACK\_HAVE\_LAPACK, 565
  - \_\_FFLASFFPACK\_HAVE\_LIMITS\_H, 565
  - \_\_FFLASFFPACK\_HAVE\_LITTLE\_ENDIAN, 565
  - \_\_FFLASFFPACK\_HAVE\_PTHREAD\_H, 565
  - \_\_FFLASFFPACK\_HAVE\_STDDEF\_H, 565
  - \_\_FFLASFFPACK\_HAVE\_STDINT\_H, 566
  - \_\_FFLASFFPACK\_HAVE\_STDIO\_H, 566
  - \_\_FFLASFFPACK\_HAVE\_STDLIB\_H, 566
  - \_\_FFLASFFPACK\_HAVE\_STRINGS\_H, 566
  - \_\_FFLASFFPACK\_HAVE\_STRING\_H, 566
  - \_\_FFLASFFPACK\_HAVE\_SYS\_STAT\_H, 566
  - \_\_FFLASFFPACK\_HAVE\_SYS\_TIME\_H, 566
  - \_\_FFLASFFPACK\_HAVE\_SYS\_TYPES\_H, 566
  - \_\_FFLASFFPACK\_HAVE\_UNISTD\_H, 566
  - \_\_FFLASFFPACK\_LT\_OBJDIR, 566
  - \_\_FFLASFFPACK\_OPENBLAS\_NUM\_THREADS, 566

- \_\_FFLASFFPACK\_PACKAGE, 566
- \_\_FFLASFFPACK\_PACKAGE\_BUGREPORT, 566
- \_\_FFLASFFPACK\_PACKAGE\_NAME, 566
- \_\_FFLASFFPACK\_PACKAGE\_STRING, 566
- \_\_FFLASFFPACK\_PACKAGE\_TARNAME, 567
- \_\_FFLASFFPACK\_PACKAGE\_URL, 567
- \_\_FFLASFFPACK\_PACKAGE\_VERSION, 567
- \_\_FFLASFFPACK\_SIZEOF\_CHAR, 567
- \_\_FFLASFFPACK\_SIZEOF\_INT, 567
- \_\_FFLASFFPACK\_SIZEOF\_LONG, 567
- \_\_FFLASFFPACK\_SIZEOF\_LONG\_LONG, 567
- \_\_FFLASFFPACK\_SIZEOF\_SHORT, 567
- \_\_FFLASFFPACK\_SIZEOF\_\_INT64\_T, 567
- \_\_FFLASFFPACK\_STDC\_HEADERS, 567
- \_\_FFLASFFPACK\_USE\_OPENMP, 567
- \_\_FFLASFFPACK\_VERSION, 567
- HAVE\_BLAS, 561
- HAVE\_CBLAS, 561
- HAVE\_CXX11, 561
- HAVE\_DLFCN\_H, 562
- HAVE\_FLOAT\_H, 562
- HAVE\_INT128, 562
- HAVE\_INTPTR\_T, 562
- HAVE\_LAPACK, 562
- HAVE\_LIMITS\_H, 562
- HAVE\_LITTLE\_ENDIAN, 562
- HAVE\_PTHREAD\_H, 562
- HAVE\_STDDEF\_H, 562
- HAVE\_STDINT\_H, 562
- HAVE\_STDIO\_H, 562
- HAVE\_STDLIB\_H, 562
- HAVE\_STRING\_H, 562
- HAVE\_STRINGS\_H, 562
- HAVE\_SYS\_STAT\_H, 562
- HAVE\_SYS\_TIME\_H, 563
- HAVE\_SYS\_TYPES\_H, 563
- HAVE\_UNISTD\_H, 563
- LT\_OBJDIR, 563
- OPENBLAS\_NUM\_THREADS, 563
- PACKAGE, 563
- PACKAGE\_BUGREPORT, 563
- PACKAGE\_NAME, 563
- PACKAGE\_STRING, 563
- PACKAGE\_TARNAME, 563
- PACKAGE\_URL, 563
- PACKAGE\_VERSION, 563
- SIZEOF\_\_INT64\_T, 564
- SIZEOF\_CHAR, 563
- SIZEOF\_INT, 563
- SIZEOF\_LONG, 563
- SIZEOF\_LONG\_LONG, 564
- SIZEOF\_SHORT, 564
- STDC\_HEADERS, 564
- USE\_OPENMP, 564
- VERSION, 564
- Configuring and Installing FFLAS-FFPACK, 2
- CONST
  - fflas\_simd.h, 616
- ConstElement\_ptr
  - benchmark-fgemv-rns.C, 532
  - rns\_double, 465
  - rns\_double\_extended, 476
  - RNSInteger< RNS >, 480
  - RNSIntegerMod< RNS >, 484
- CONSTREFERENCE
  - parallel.h, 778
- convert
  - rns\_double, 466, 467
  - rns\_double\_extended, 477
  - RNSInteger< RNS >, 481
  - RNSIntegerMod< RNS >, 486
- convert\_transpose
  - rns\_double, 466
- ConvertTo< T >, 414
- COO
  - FFLAS, 76
- Coo
  - Coo< Field >, 416
  - Coo< ValT, IdxT >, 415, 417, 418
- coo
  - HelperFlag, 445
- Coo< Field >, 416
  - col, 417
  - Coo, 416
  - deleted, 417
  - operator=, 416
  - row, 417
  - val, 417
- Coo< ValT, IdxT >, 414, 417
  - col, 415, 418
  - Coo, 415, 417, 418
  - operator=, 415, 418
  - row, 415, 418
  - Self, 414, 417
  - val, 415, 418
- coo.h, 632
- coo\_spmv.inl, 632
  - \_\_FFLASFFPACK\_fflas\_sparse\_coo\_spmv\_INL, 633
- coo\_spmv.inl, 633
  - \_\_FFLASFFPACK\_fflas\_sparse\_coo\_spmv\_INL, 634
- coo\_utils.inl, 634
  - \_\_FFLASFFPACK\_fflas\_sparse\_coo\_utils\_INL, 634
- COO\_ZO
  - FFLAS, 76
- CooMat< Field >, 418
  - \_coo16, 419
  - \_coo16\_zo, 419
  - \_coo32, 419
  - \_coo32\_zo, 419
  - \_coo64, 419
  - \_coo64\_zo, 419
- Copying and Licence, 2
- count\_nonconst\_lvalue\_reference< T >, 419



- CROUT
  - ffpack\_pluq.inl, [686](#)
- CSC
  - FFLAS, [76](#)
- CSC\_ZO
  - FFLAS, [76](#)
- CSR
  - FFLAS, [76](#)
- csr
  - HelperFlag, [445](#)
- csr.h, [634](#)
- CSR\_HYB
  - FFLAS, [76](#)
- csr\_hyb.h, [639](#)
- csr\_hyb\_pspmm.inl, [639](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_pspmm\_INL, [640](#)
- csr\_hyb\_pspmv.inl, [640](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_pspmv\_INL, [640](#)
- csr\_hyb\_spm্ম.inl, [641](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_spm্ম\_INL, [641](#)
- csr\_hyb\_spmv.inl, [641](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_spmv\_INL, [641](#)
- csr\_hyb\_utils.inl, [642](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_utils\_INL, [642](#)
- csr\_pspmm.inl, [635](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_pspmm\_INL, [636](#)
- csr\_pspmv.inl, [636](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_pspmv\_INL, [636](#)
- csr\_spm্ম.inl, [636](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_spm্ম\_INL, [637](#)
- csr\_spmv.inl, [638](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_spmv\_INL, [638](#)
- csr\_utils.inl, [638](#)
- CSR\_ZO
  - FFLAS, [76](#)
- CsrMat< Field >, [419](#)
  - \_csr16, [420](#)
  - \_csr16\_zo, [420](#)
  - \_csr32, [420](#)
  - \_csr32\_zo, [420](#)
  - \_csr64, [420](#)
  - \_csr64\_zo, [420](#)
- ctz
  - bit\_manipulation.h, [787](#)
- CUBE
  - benchmark-checkers.C, [525](#)
  - benchmark-fsytrf.C, [540](#)
  - benchmark-ftsri.C, [542](#)
  - benchmark-inverse.C, [543](#)
  - benchmark-lqup.C, [544](#)
  - benchmark-pluq.C, [545](#)
  - benchmark-wino.C, [547](#)
  - charpoly.C, [518](#)
  - fsyrk.C, [519](#)
  - fsytrf.C, [520](#)
  - ftsri.C, [520](#)
  - pluq.C, [521](#)
  - cuda.C, [796](#)
  - main, [796](#)
- current
  - ForStrategy1D< blocksize\_t, Cut, Param >, [433](#)
  - ForStrategy2D< blocksize\_t, Cut, Param >, [437](#)
- Cut
  - Parallel< C, P >, [459](#)
- cyclic\_shift\_col
  - FFPACK, [360](#), [363](#)
- cyclic\_shift\_col\_modular\_double
  - ffpack.C, [734](#)
  - ffpack\_c.h, [755](#)
- cyclic\_shift\_mathPerm
  - FFPACK, [359](#)
  - ffpack.C, [733](#)
  - ffpack\_c.h, [755](#)
- cyclic\_shift\_row
  - FFPACK, [359](#), [362](#)
- cyclic\_shift\_row\_col
  - FFPACK, [359](#), [362](#)
- cyclic\_shift\_row\_modular\_double
  - ffpack.C, [733](#)
  - ffpack\_c.h, [755](#)
- Danilevski
  - FFPACK, [349](#)
  - FFPACK::Protected, [393](#)
- dasum\_
  - config-blas.h, [557](#)
- data
  - Argument, [398](#)
- daxpy\_
  - config-blas.h, [556](#)
- dcopy\_
  - config-blas.h, [558](#)
- ddot\_
  - config-blas.h, [556](#)
- debug.h, [787](#)
  - FFLASFFPACK\_abort, [788](#)
  - FFLASFFPACK\_check, [788](#)
- DeCompressRows
  - FFPACK::Protected, [395](#)
- DeCompressRowsQA
  - FFPACK::Protected, [396](#)
- DeCompressRowsQK
  - FFPACK::Protected, [395](#)
- DefaultBoundedTag, [420](#)
- DefaultTag, [420](#)
- delayed
  - RNSIntegerMod< RNS >, [484](#)
- DelayedField



- MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-  
Trait >, [452](#)
- delayedField
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-  
Trait >, [456](#)
- DelayedField\_t
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-  
Trait >, [452](#)
- DelayedTag, [421](#)
- deleted
  - Coo< Field >, [417](#)
- DENSE\_THRESHOLD
  - fflas\_sparse.h, [629](#)
- denseCols
  - StatsMatrix, [504](#)
- denseRows
  - StatsMatrix, [503](#)
- Det
  - FFPACK, [326](#), [327](#), [350](#), [369](#), [370](#)
- det.C, [548](#)
  - main, [548](#)
- Det\_modular\_double
  - ffpack.C, [743](#)
  - ffpack\_c.h, [763](#)
- deviationCol
  - StatsMatrix, [503](#)
- deviationColDifference
  - StatsMatrix, [503](#)
- deviationRow
  - StatsMatrix, [502](#)
- deviationRowDifference
  - StatsMatrix, [503](#)
- DFelt
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-  
Trait >, [452](#)
- dgemm\_
  - config-blas.h, [560](#)
  - fblas.C, [796](#)
- dgemv\_
  - config-blas.h, [557](#)
- dger\_
  - config-blas.h, [558](#)
- div
  - ScalFunctions< Element >, [493](#)
- DivideAndConquer, [421](#)
- dnrm2\_
  - config-blas.h, [557](#)
- DNS\_BIN\_VER
  - read\_sparse.h, [653](#)
- doApplyS
  - FFPACK, [355](#)
- doApplyT
  - FFPACK, [356](#)
- doBenchs
  - Bench< Elt >, [401](#)
- doTests
  - Test< Elt >, [507](#)
- DotProdBoundClassic
  - FFLAS::Protected, [212](#)
- DOUBLE\_TO\_FLOAT\_CROSSOVER
  - fflas.h, [570](#)
  - winograd.C, [522](#)
- dscal\_
  - config-blas.h, [558](#)
- dtrmm\_
  - config-blas.h, [559](#)
- dtrsm\_
  - config-blas.h, [559](#)
- DynamicPeeling
  - FFLAS::Protected, [216](#)
- DynamicPeeling2
  - FFLAS::Protected, [216](#)
- EFFGFF
  - benchmark-dsytrf.C, [528](#)
- Element
  - FieldSimd< \_Field >, [426](#)
  - readMyMachineType< Field, T >, [462](#)
  - rns\_double, [465](#)
  - rns\_double\_extended, [476](#)
  - RNSInteger< RNS >, [480](#)
  - RNSIntegerMod< RNS >, [484](#)
  - TestOneMethod< Simd >, [509](#)
- Element\_ptr
  - benchmark-fgemm-rns.C, [532](#)
  - readMyMachineType< Field, T >, [462](#)
  - rns\_double, [465](#)
  - rns\_double\_extended, [476](#)
  - RNSInteger< RNS >, [480](#)
  - RNSIntegerMod< RNS >, [484](#)
- elements
  - array< T >, [399](#)
  - vector< T >, [516](#)
- ElementTraits< Element >, [421](#)
  - value, [421](#)
- ELL
  - FFLAS, [76](#)
- ell
  - HelperFlag, [445](#)
- ell.h, [642](#)
- ell\_pspmm.inl, [643](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_pspmm\_INL,  
[643](#)
- ell\_pspmv.inl, [643](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_pspmv\_INL,  
[644](#)
- ELL\_simd
  - FFLAS, [76](#)
- ell\_simd.h, [646](#)
- ell\_simd\_pspmv.inl, [647](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_simd\_pspmv\_INL,  
[648](#)
- ell\_simd\_spmv.inl, [648](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_simd\_spmv\_INL,  
[648](#)
- ell\_simd\_utils.inl, [649](#)

- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_simd\_utils\_INL, 649
- ELL\_simd\_ZO
  - FFLAS, 76
- ell\_spmv.inl, 644
  - \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_spmv\_INL, 645
- ell\_spmv.inl, 645
  - \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_spmv\_INL, 646
- ell\_utils.inl, 646
  - \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_utils\_INL, 646
- ELL\_ZO
  - FFLAS, 76
- ElIMat< Field >, 421
  - \_ell16, 422
  - \_ell16\_zo, 422
  - \_ell32, 422
  - \_ell32\_zo, 422
  - \_ell64, 422
  - \_ell64\_zo, 422
- Elt\_ptr
  - Bench< Elt >, 400
  - Test< Elt >, 506
- ENABLE\_ALL\_CHECKINGS
  - benchmark-checkers.C, 525
  - ffpack\_ftrtr.inl, 680
  - test-fdot.C, 806
  - test-fgemm-check.C, 807
  - test-fsyr2k.C, 818
  - test-fsyrk.C, 820
  - test-ftrmv.C, 824
  - test-ftrsm-check.C, 825
  - test-ftrsm.C, 826
  - test-ftrssyr2k.C, 827
  - test-ftrstr.C, 828
  - test-ftrsv.C, 829
  - test-ftrtri.C, 830
  - test-invert-check.C, 832
  - test-pluq-check.C, 841
- ENABLE\_CHECKER\_charpoly
  - test-charpoly-check.C, 798
- ENABLE\_CHECKER\_Det
  - test-det-check.C, 801
- ENABLE\_CHECKER\_fgemm
  - test-fgemm.C, 809
- enable\_if\_no\_simd\_t
  - Bench< Elt >, 400
  - Test< Elt >, 506
- enable\_if\_simd128\_t
  - Bench< Elt >, 401
  - Test< Elt >, 506
- enable\_if\_simd256\_t
  - Bench< Elt >, 401
  - Test< Elt >, 506
- enable\_if\_simd512\_t
  - Bench< Elt >, 401
- Test< Elt >, 507
- enable\_if\_t
  - Bench< Elt >, 400
  - Test< Elt >, 506
  - TestOneMethod< Simd >, 509
- end
  - ForStrategy1D< blocksize\_t, Cut, Param >, 433
- END\_OF\_ARGUMENTS
  - args-parser.h, 785
- END\_PARALLEL\_MAIN
  - parallel.h, 778
- eq
  - ScalFunctions< Element >, 494
- eval\_func\_on\_array
  - test-simd.C, 847
- evaluate\_scalar\_method
  - TestOneMethod< Simd >, 509, 510
- evaluate\_simd\_method
  - TestOneMethod< Simd >, 510
- example
  - Argument, 398
- ExpandBlockBiDiagonalToBruhat
  - FFPACK, 343
- expandLCRE
  - FFPACK, 347
- F
  - Bench< Elt >, 402
  - Test< Elt >, 508
- fadd
  - FFLAS, 77, 78, 80, 82, 164, 171, 172
  - FFLAS::details, 199, 200
- fadd\_1\_modular\_double
  - fflas\_c.h, 701
  - fflas\_lvl1.C, 720
- fadd\_2\_modular\_double
  - fflas\_c.h, 704
  - fflas\_lvl2.C, 725
- faddin
  - FFLAS, 78, 81, 164, 172
- faddin\_1\_modular\_double
  - fflas\_c.h, 701
  - fflas\_lvl1.C, 721
- faddin\_2\_modular\_double
  - fflas\_c.h, 705
  - fflas\_lvl2.C, 725
- Failure, 422
  - \_errorStream, 424
  - Failure, 423
  - operator(), 423
  - print, 423
  - setErrorStream, 423
- failure
  - FFPACK, 375
- FailureCharpolyCheck, 424
- FailureDetCheck, 424
- FailureFgemmCheck, 424
- FailureInvertCheck, 424
- FailurePLUQCheck, 424

- FailureTrsmCheck, [424](#)
- fassign
  - FFLAS, [82–84](#), [160](#), [164](#)
- fassign\_1\_modular\_double
  - fflas\_c.h, [699](#)
  - fflas\_lvl1.C, [719](#)
- fassign\_2\_modular\_double
  - fflas\_c.h, [701](#)
  - fflas\_lvl2.C, [722](#)
- faxpby
  - FFLAS, [132](#), [139](#)
- faxpy
  - FFLAS, [84](#), [85](#), [162](#), [169](#)
  - FFLAS::details, [201](#)
- faxpy\_1\_modular\_double
  - fflas\_c.h, [700](#)
  - fflas\_lvl1.C, [720](#)
- faxpy\_2\_modular\_double
  - fflas\_c.h, [704](#)
  - fflas\_lvl2.C, [724](#)
- fblas.C, [796](#)
  - \_\_FFLASFFPACK\_CONFIGURATION, [796](#)
  - dgemm\_, [796](#)
  - main, [796](#)
- fconvert
  - FFLAS, [129](#), [137](#), [158](#)
- fconvert\_rns
  - FFLAS, [156](#), [157](#)
- fconvert\_trans\_rns
  - FFLAS, [156](#)
- fdot
  - FFLAS, [86–88](#), [133](#), [162](#), [181](#)
- fdot\_1\_modular\_double
  - fflas\_c.h, [700](#)
  - fflas\_lvl1.C, [720](#)
- fequal
  - FFLAS, [132](#), [136](#), [160](#), [165](#)
- fequal\_1\_modular\_double
  - fflas\_c.h, [699](#)
  - fflas\_lvl1.C, [719](#)
- fequal\_2\_modular\_double
  - fflas\_c.h, [702](#)
  - fflas\_lvl2.C, [722](#)
- FFLAS, [42](#), [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\_ftsm, [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](#), [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, [156](#), [157](#)
  - fconvert\_trans\_rns, [156](#)
  - fdot, [86–88](#), [133](#), [162](#), [181](#)
  - fequal, [132](#), [136](#), [160](#), [165](#)
  - FFLAS\_BASE, [75](#)
  - fflas\_delete, [155](#), [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, 158, 167
- finit\_rns, 155, 156
- finit\_trans\_rns, 155
- fiszero, 131, 136, 160, 166
- fmove, 139, 170
- fneg, 130, 138, 159, 168
- fnegin, 130, 138, 159, 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
- ftranspose, 154
- 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, 851
  - main, 851
- fflas-101\_3.C, 851
  - main, 851
- FFLAS-FFPACK, 41
- FFLAS-FFPACK Documentation., 1
- FFLAS-FFPACK fields, 43
- fflas-ffpack-config.h, 567
  - GCC\_VERSION, 568
- fflas-ffpack-default-thresholds.h, 568
  - \_\_FFLASFFPACK\_ARITHPROG\_THRESHOLD, 569
  - \_\_FFLASFFPACK\_CHARPOLY\_Danilevskii\_LUKrylov\_THRESHOLD, 568
  - \_\_FFLASFFPACK\_CHARPOLY\_LUKrylov\_ArithProg\_THRESHOLD, 568
  - \_\_FFLASFFPACK\_FSYRK\_THRESHOLD, 569
  - \_\_FFLASFFPACK\_FSYTRF\_THRESHOLD, 569
  - \_\_FFLASFFPACK\_FTRTRI\_THRESHOLD, 569
  - \_\_FFLASFFPACK\_PLUQ\_THRESHOLD, 568
  - \_\_FFLASFFPACK\_WINOTHRESHOLD, 568
  - \_\_FFLASFFPACK\_WINOTHRESHOLD\_BAL, 568
  - \_\_FFLASFFPACK\_WINOTHRESHOLD\_BAL\_FLT, 568
  - \_\_FFLASFFPACK\_WINOTHRESHOLD\_FLT, 568
- fflas-ffpack-thresholds.h, 569
- fflas-ffpack.doxy, 569
- fflas-ffpack.h, 569
- fflas.doxy, 569
- fflas.h, 569
  - DOUBLE\_TO\_FLOAT\_CROSSOVER, 570
  - WINOTHRESHOLD, 570
- 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
  - fscale, 202, 203
  - fscalein, 202, 203
  - gebp, 205
  - igebb11, 204
  - igebb14, 204
  - igebb21, 204
  - igebb24, 203
  - igebb41, 204
  - igebb44, 203
  - igebp, 205
  - pack\_lhs, 205
  - pack\_rhs, 205
  - FFLAS::details\_spmv, 206
  - FFLAS::FieldCategories, 206
  - FFLAS::FieldCategories, 206
  - FFLAS::MMHelperAlgo, 207
  - FFLAS::ModeCategories, 207
  - FFLAS::ParSeqHelper, 208
  - FFLAS::Protected, 208
    - computeFactorClassic, 211
    - DotProdBoundClassic, 212
    - DynamicPeeling, 216
    - DynamicPeeling2, 216
    - fgemm\_convert, 212
    - fgemv\_convert, 217
    - fger\_convert, 218
    - fsquareCommon, 215
    - fsyrk\_convert, 218
    - igemm, 221
    - igemm\_colmajor, 221
    - MatF2MatD\_Triangular, 222
    - MatF2MatFI\_Triangular, 222
    - min\_types, 219, 220
    - NeedDoublePreAddReduction, 214
    - NeedPreAddReduction, 213
    - NeedPreAxyReduction, 219
    - NeedPreScalReduction, 218, 219
    - NeedPreSubReduction, 213
    - ScalAndReduce, 214, 218
    - TRSMBound, 212
    - unfit, 220, 221
    - WinogradCalc, 217
    - WinogradSteps, 216
    - WinogradThreshold, 215
  - FFLAS::sell\_details, 222
  - FFLAS::sparse\_details, 223
    - fspmm, 229–231
    - fspmm\_dispatch, 228, 229
    - fspmv, 226–228, 236
    - fspmv\_dispatch, 226
    - init\_y, 226
    - pfspmm, 232–234
    - pfspmm\_dispatch, 231, 232
    - pfspmv, 234, 235
  - FFLAS::sparse\_details\_impl, 237

- fspmm, [245](#), [252](#), [253](#), [258](#), [259](#), [263](#), [272](#)
- fspmm\_mone, [246](#), [254](#), [264](#)
- fspmm\_mone\_simd\_aligned, [247](#), [254](#), [265](#)
- fspmm\_mone\_simd\_unaligned, [247](#), [255](#), [265](#)
- fspmm\_one, [246](#), [253](#), [264](#)
- fspmm\_one\_simd\_aligned, [247](#), [254](#), [265](#)
- fspmm\_one\_simd\_unaligned, [247](#), [254](#), [265](#)
- fspmm\_simd\_aligned, [246](#), [253](#)
- fspmm\_simd\_unaligned, [246](#), [253](#)
- fspmv, [248](#), [255](#), [259](#), [266](#), [268](#), [269](#), [273](#), [275](#)
- fspmv\_mone, [248](#), [249](#), [256](#), [266](#), [267](#), [270](#), [276](#), [277](#)
- fspmv\_mone\_simd, [270](#), [276](#)
- fspmv\_one, [248](#), [249](#), [255](#), [256](#), [266](#), [267](#), [269](#), [270](#), [276](#)
- fspmv\_one\_simd, [270](#), [276](#)
- fspmv\_simd, [269](#), [275](#)
- pfspmm, [249](#), [256](#), [257](#), [259–261](#), [271](#)
- pfspmm\_mone, [250](#)
- pfspmm\_one, [250](#)
- pfspmm\_zo, [261](#)
- pfspmv, [251](#), [258](#), [261](#), [262](#), [267](#), [271–274](#)
- pfspmv\_mone, [252](#), [262](#), [263](#), [268](#), [274](#)
- pfspmv\_one, [251](#), [252](#), [262](#), [268](#), [274](#)
- pfspmv\_task, [251](#)
- FFLAS::StrategyParameter, [277](#)
- FFLAS::StructureHelper, [277](#)
- FFLAS::vectorised, [278](#)
  - add, [280](#)
  - addp, [279](#)
  - axpy, [280](#)
  - modp, [282](#)
  - reduce, [281](#), [282](#)
  - scalp, [282](#), [283](#)
  - sub, [280](#)
  - subp, [280](#)
  - VEC\_ADD, [279](#)
  - VEC\_SUB, [279](#)
- FFLAS::vectorised::unswitch, [283](#)
  - axpy, [284](#)
  - modp, [284](#), [285](#)
  - scalp, [285](#)
- fflas\_101.C, [852](#)
  - main, [852](#)
- fflas\_101\_lvl1.C, [852](#)
  - main, [852](#)
- FFLAS\_BASE
  - FFLAS, [75](#)
- fflas\_bounds.inl, [571](#)
  - \_\_FFLASFFPACK\_fflas\_bounds\_INL, [571](#)
  - FFLAS\_INT\_TYPE, [571](#)
- fflas\_c.h, [695](#)
  - fadd\_1\_modular\_double, [701](#)
  - fadd\_2\_modular\_double, [704](#)
  - faddin\_1\_modular\_double, [701](#)
  - faddin\_2\_modular\_double, [705](#)
  - fassign\_1\_modular\_double, [699](#)
  - fassign\_2\_modular\_double, [701](#)
  - faxpy\_1\_modular\_double, [700](#)
  - faxpy\_2\_modular\_double, [704](#)
  - fdot\_1\_modular\_double, [700](#)
  - fequal\_1\_modular\_double, [699](#)
  - fequal\_2\_modular\_double, [702](#)
  - FFLAS\_C\_BASE, [698](#)
  - FFLAS\_C\_DIAG, [698](#)
  - FFLAS\_C\_ORDER, [697](#)
  - FFLAS\_C\_SIDE, [698](#)
  - FFLAS\_C\_TRANSPOSE, [697](#)
  - FFLAS\_C\_UPLO, [697](#)
  - FFLAS\_COMPILED, [697](#)
  - FflasColMajor, [697](#)
  - FflasDouble, [698](#)
  - FflasFloat, [698](#)
  - FflasGeneric, [698](#)
  - FflasLeft, [698](#)
  - FflasLower, [698](#)
  - FflasNonUnit, [698](#)
  - FflasNoTrans, [697](#)
  - FflasRight, [698](#)
  - FflasRowMajor, [697](#)
  - FflasTrans, [697](#)
  - FflasUnit, [698](#)
  - FflasUpper, [698](#)
  - fgemm\_3\_modular\_double, [706](#)
  - fgemv\_2\_modular\_double, [705](#)
  - fger\_2\_modular\_double, [705](#)
  - fidentity\_2\_modular\_double, [702](#)
  - fiszero\_1\_modular\_double, [699](#)
  - fiszero\_2\_modular\_double, [702](#)
  - fmove\_2\_modular\_double, [704](#)
  - fneg\_1\_modular\_double, [699](#)
  - fneg\_2\_modular\_double, [703](#)
  - fnegin\_1\_modular\_double, [699](#)
  - fnegin\_2\_modular\_double, [703](#)
  - freduce\_1\_modular\_double, [698](#)
  - freduce\_2\_modular\_double, [703](#)
  - freducein\_1\_modular\_double, [698](#)
  - freducein\_2\_modular\_double, [702](#)
  - fscal\_1\_modular\_double, [700](#)
  - fscal\_2\_modular\_double, [703](#)
  - fscalin\_1\_modular\_double, [700](#)
  - fscalin\_2\_modular\_double, [703](#)
  - fsquare\_3\_modular\_double, [707](#)
  - fsub\_1\_modular\_double, [701](#)
  - fsub\_2\_modular\_double, [704](#)
  - fsubin\_1\_modular\_double, [701](#)
  - fsubin\_2\_modular\_double, [704](#)
  - fswap\_1\_modular\_double, [700](#)
  - ftmm\_3\_modular\_double, [706](#)
  - ftsm\_3\_modular\_double, [706](#)
  - ftsv\_2\_modular\_double, [705](#)
  - fzero\_1\_modular\_double, [699](#)
  - fzero\_2\_modular\_double, [702](#)
- FFLAS\_C\_BASE
  - fflas\_c.h, [698](#)
- FFLAS\_C\_DIAG

- fflas\_c.h, [698](#)
- ffpack\_c.h, [752](#)
- FFLAS\_C\_ORDER
  - fflas\_c.h, [697](#)
  - ffpack\_c.h, [752](#)
- FFLAS\_C\_SIDE
  - fflas\_c.h, [698](#)
  - ffpack\_c.h, [752](#)
- FFLAS\_C\_TRANSPOSE
  - fflas\_c.h, [697](#)
  - ffpack\_c.h, [752](#)
- FFLAS\_C\_UPLO
  - fflas\_c.h, [697](#)
  - ffpack\_c.h, [752](#)
- FFLAS\_COMPILED
  - fflas\_c.h, [697](#)
  - ffpack\_inst.C, [769](#)
  - ffpack\_inst.h, [770](#)
- fflas\_const\_cast
  - FFPACK, [362](#), [375](#)
- fflas\_delete
  - FFLAS, [155](#), [188](#)
- FFLAS\_DIAG
  - FFLAS, [75](#)
- FFLAS\_ELT
  - fflas\_L1\_inst.C, [707](#), [708](#)
  - fflas\_L1\_inst.h, [708](#), [709](#)
  - fflas\_L2\_inst.C, [711](#)
  - fflas\_L2\_inst.h, [712](#)
  - fflas\_L3\_inst.C, [714](#), [715](#)
  - fflas\_L3\_inst.h, [715](#), [716](#)
  - ffpack\_inst.C, [769](#)
  - ffpack\_inst.h, [770](#)
- fflas\_enum.h, [571](#)
- fflas\_fadd.h, [572](#)
- fflas\_fadd.inl, [573](#)
  - \_\_FFLASFFPACK\_fadd\_INL, [574](#)
- fflas\_fassign.h, [574](#)
- fflas\_fassign.inl, [575](#)
  - \_\_FFLASFFPACK\_fassign\_INL, [575](#)
- fflas\_faxpy.inl, [575](#)
  - \_\_FFLASFFPACK\_faxpy\_INL, [576](#)
- fflas\_fdot.inl, [576](#)
  - \_\_FFLASFFPACK\_fdot\_INL, [577](#)
- fflas\_fgemm.inl, [577](#)
  - \_\_FFLASFFPACK\_fgemm\_INL, [579](#)
- fflas\_fgemv.inl, [586](#)
  - \_\_FFLASFFPACK\_fgemv\_INL, [587](#)
- fflas\_fgemv\_mp.inl, [588](#)
  - \_\_FFLASFFPACK\_fgemv\_mp\_INL, [588](#)
- fflas\_fger.inl, [588](#)
  - \_\_FFLASFFPACK\_fger\_INL, [589](#)
- fflas\_fger\_mp.inl, [589](#)
  - \_\_FFPACK\_fger\_mp\_INL, [590](#)
- FFLAS\_FIELD
  - fflas\_L1\_inst.C, [707](#), [708](#)
  - fflas\_L1\_inst.h, [708](#), [709](#)
  - fflas\_L2\_inst.C, [711](#)
- fflas\_L2\_inst.h, [712](#)
- fflas\_L3\_inst.C, [714](#), [715](#)
- fflas\_L3\_inst.h, [715](#), [716](#)
- ffpack\_inst.C, [769](#)
- ffpack\_inst.h, [770](#)
- FFLAS\_FORMAT
  - FFLAS, [76](#)
- fflas\_freduce.h, [590](#)
- fflas\_freduce.inl, [591](#)
  - \_\_FFLASFFPACK\_fflas\_freduce\_INL, [593](#)
  - FFLASFFPACK\_COPY\_REDUCE, [593](#)
- fflas\_freduce\_mp.inl, [593](#)
  - \_\_FFLASFFPACK\_fflas\_freduce\_mp\_INL, [593](#)
- fflas\_freivalds.inl, [593](#)
  - \_\_FFLASFFPACK\_freivalds\_INL, [593](#)
- fflas\_fscal.h, [594](#)
- fflas\_fscal.inl, [594](#)
  - \_\_FFLASFFPACK\_fscal\_INL, [595](#)
- fflas\_fscal\_mp.inl, [595](#)
  - \_\_FFLASFFPACK\_fscal\_mp\_INL, [596](#)
- fflas\_fsyr2k.inl, [596](#)
  - \_\_FFLASFFPACK\_fflas\_fsyr2k\_INL, [596](#)
- fflas\_fsyrk.inl, [597](#)
  - \_\_FFLASFFPACK\_fflas\_fsyrk\_INL, [598](#)
- fflas\_fsyrk\_strassen.inl, [598](#)
  - \_\_FFLASFFPACK\_fflas\_fsyrk\_strassen\_INL, [599](#)
- fflas\_ftrmm.inl, [599](#)
  - \_\_FFLASFFPACK\_ftrmm\_INL, [600](#)
- fflas\_ftrsm.inl, [600](#)
  - \_\_FFLASFFPACK\_ftrsm\_INL, [600](#)
- fflas\_ftrsm\_mp.inl, [601](#)
  - \_\_FFPACK\_ftrsm\_mp\_INL, [601](#)
- fflas\_ftrsv.inl, [601](#)
  - \_\_FFLASFFPACK\_ftrsv\_INL, [602](#)
- fflas\_helpers.inl, [602](#)
  - \_\_FFLASFFPACK\_fflas\_fflas\_mmhelper\_INL, [603](#)
- FFLAS\_INT\_TYPE
  - fflas\_bounds.inl, [571](#)
- fflas\_intrinsic.h, [788](#)
- fflas\_io.h, [788](#)
- fflas\_L1\_inst.C, [707](#)
  - \_\_FFLAS\_L1\_INST\_C, [707](#)
- FFLAS\_ELT, [707](#), [708](#)
- FFLAS\_FIELD, [707](#), [708](#)
- INST\_OR\_DECL, [707](#)
- fflas\_L1\_inst.h, [708](#)
  - FFLAS\_ELT, [708](#), [709](#)
  - FFLAS\_FIELD, [708](#), [709](#)
  - INST\_OR\_DECL, [708](#)
- fflas\_L1\_inst\_implem.inl, [709](#)
- fflas\_L2\_inst.C, [710](#)
  - \_\_FFLAS\_L2\_INST\_C, [711](#)
- FFLAS\_ELT, [711](#)
- FFLAS\_FIELD, [711](#)
- INST\_OR\_DECL, [711](#)
- fflas\_L2\_inst.h, [711](#)
  - FFLAS\_ELT, [712](#)
  - FFLAS\_FIELD, [712](#)



INST\_OR\_DECL, 712  
 fflas\_L2\_inst\_implement.inl, 712  
 fflas\_L3\_inst.C, 714  
     \_\_FFLAS\_L3\_INST\_C, 714  
     FFLAS\_ELT, 714, 715  
     FFLAS\_FIELD, 714, 715  
     INST\_OR\_DECL, 714  
 fflas\_L3\_inst.h, 715  
     FFLAS\_ELT, 715, 716  
     FFLAS\_FIELD, 715, 716  
     INST\_OR\_DECL, 715  
 fflas\_L3\_inst\_implement.inl, 716  
     \_\_FFLAS\_TRSM\_READONLY, 717  
 fflas\_level1.inl, 606  
     \_\_FFLASFFPACK\_fflas\_fflas\_level1\_INL, 608  
 fflas\_level2.inl, 609  
     \_\_FFLASFFPACK\_fflas\_fflas\_level2\_INL, 611  
 fflas\_level3.inl, 611  
     \_\_FFLASFFPACK\_fflas\_fflas\_level3\_INL, 614  
     \_\_FFLAS\_TRSM\_READONLY, 614  
 fflas\_lvl1.C, 717  
     fadd\_1\_modular\_double, 720  
     faddin\_1\_modular\_double, 721  
     fassign\_1\_modular\_double, 719  
     faxpy\_1\_modular\_double, 720  
     fdot\_1\_modular\_double, 720  
     fequal\_1\_modular\_double, 719  
     fiszero\_1\_modular\_double, 719  
     fneg\_1\_modular\_double, 718  
     fnegin\_1\_modular\_double, 718  
     freduce\_1\_modular\_double, 718  
     freducein\_1\_modular\_double, 718  
     fscal\_1\_modular\_double, 719  
     fscalin\_1\_modular\_double, 719  
     fsub\_1\_modular\_double, 720  
     fsubin\_1\_modular\_double, 721  
     fswap\_1\_modular\_double, 720  
     fzero\_1\_modular\_double, 718  
 fflas\_lvl2.C, 721  
     fadd\_2\_modular\_double, 725  
     faddin\_2\_modular\_double, 725  
     fassign\_2\_modular\_double, 722  
     faxpy\_2\_modular\_double, 724  
     fequal\_2\_modular\_double, 722  
     fgemv\_2\_modular\_double, 725  
     fger\_2\_modular\_double, 726  
     fidentity\_2\_modular\_double, 723  
     fiszero\_2\_modular\_double, 723  
     fmove\_2\_modular\_double, 724  
     fneg\_2\_modular\_double, 724  
     fnegin\_2\_modular\_double, 723  
     freduce\_2\_modular\_double, 723  
     freducein\_2\_modular\_double, 723  
     fscal\_2\_modular\_double, 724  
     fscalin\_2\_modular\_double, 724  
     fsub\_2\_modular\_double, 725  
     fsubin\_2\_modular\_double, 725  
     ftrsv\_2\_modular\_double, 726  
     fzero\_2\_modular\_double, 722  
 fflas\_lvl3.C, 726  
     fgemm\_3\_modular\_double, 727  
     fsquare\_3\_modular\_double, 728  
     ftrmm\_3\_modular\_double, 727  
     ftrsm\_3\_modular\_double, 727  
 fflas\_memory.h, 789  
 fflas\_new  
     FFLAS, 155, 156, 188  
 FFLAS\_ORDER  
     FFLAS, 74  
 fflas\_pfgemm.inl, 614  
     \_\_FFLASFFPACK\_DIMKPENALTY, 614  
     \_\_FFLASFFPACK\_SEQPARTHRESHOLD, 614  
     \_\_FFLASFFPACK\_fflas\_pfgemm\_INL, 614  
 fflas\_pfrsm.inl, 615  
     \_\_FFLASFFPACK\_fflas\_pfrsm\_INL, 615  
     PTRSM\_HYBRID\_THRESHOLD, 615  
 fflas\_plevel1.h, 775  
 fflas\_randommatrix.h, 790  
 FFLAS\_SIDE  
     FFLAS, 75  
 fflas\_simd.h, 615  
     CONST, 616  
     FLOAT\_MOD, 617  
     INLINE, 616  
     NORML\_MOD, 616  
     PURE, 616  
     Simd, 617  
     SIMD\_INT, 616  
 fflas\_sparse.C, 728  
 fflas\_sparse.h, 625  
     \_\_FFLASFFPACK\_CACHE\_LINE\_SIZE, 629  
     assume\_aligned, 629  
     DENSE\_THRESHOLD, 629  
     index\_t, 629  
     ROUND\_DOWN, 629  
 fflas\_sparse.inl, 630  
     \_\_FFLASFFPACK\_fflas\_fflas\_sparse\_INL, 631  
 FFLAS\_TRANSPOSE  
     FFLAS, 74  
 fflas\_transpose.h, 657  
     FFLAS\_TRANSPOSE\_BLOCKSIZE, 658  
     LD, 658  
     ST, 658  
 FFLAS\_TRANSPOSE\_BLOCKSIZE  
     fflas\_transpose.h, 658  
 FFLAS\_UPLO  
     FFLAS, 75  
 FflasAuto  
     FFLAS, 76  
 FflasBinary  
     FFLAS, 76  
 FflasColMajor  
     FFLAS, 74  
     fflas\_c.h, 697  
     ffpack\_c.h, 752  
 FflasDense



- FFLAS, [76](#)
- FflasDouble
  - FFLAS, [76](#)
  - fflas\_c.h, [698](#)
- FFLASFFPACK\_abort
  - debug.h, [788](#)
- FFLASFFPACK\_check
  - debug.h, [788](#)
- FFLASFFPACK\_checkers\_fflas\_inl\_H
  - checkers\_fflas.inl, [553](#)
- FFLASFFPACK\_checkers\_ffpack\_inl\_H
  - checkers\_ffpack.inl, [554](#)
- FFLASFFPACK\_COPY\_REDUCE
  - fflas\_freduce.inl, [593](#)
- FFLASFFPACK\_PERM\_BKSIZE
  - ffpack\_permutation.inl, [685](#)
- FflasFloat
  - FFLAS, [76](#)
  - fflas\_c.h, [698](#)
- FflasGeneric
  - FFLAS, [76](#)
  - fflas\_c.h, [698](#)
- FflasLeft
  - FFLAS, [75](#)
  - fflas\_c.h, [698](#)
  - ffpack\_c.h, [753](#)
- FflasLeftTri
  - FFLAS, [75](#)
- FflasLower
  - FFLAS, [75](#)
  - fflas\_c.h, [698](#)
  - ffpack\_c.h, [752](#)
- FflasMaple
  - FFLAS, [76](#)
- FflasMath
  - FFLAS, [76](#)
- FflasNonUnit
  - FFLAS, [75](#)
  - fflas\_c.h, [698](#)
  - ffpack\_c.h, [752](#)
- FflasNoTrans
  - FFLAS, [75](#)
  - fflas\_c.h, [697](#)
  - ffpack\_c.h, [752](#)
- FflasRight
  - FFLAS, [75](#)
  - fflas\_c.h, [698](#)
  - ffpack\_c.h, [752](#)
- FflasRightTri
  - FFLAS, [75](#)
- FflasRowMajor
  - FFLAS, [74](#)
  - fflas\_c.h, [697](#)
  - ffpack\_c.h, [752](#)
- FflasSageMath
  - FFLAS, [76](#)
- FflasSMS
  - FFLAS, [76](#)
- FflasTrans
  - FFLAS, [75](#)
  - fflas\_c.h, [697](#)
  - ffpack\_c.h, [752](#)
- FflasUnit
  - FFLAS, [75](#)
  - fflas\_c.h, [698](#)
  - ffpack\_c.h, [752](#)
- FflasUpper
  - FFLAS, [75](#)
  - fflas\_c.h, [698](#)
  - ffpack\_c.h, [752](#)
- FFPACK, [43](#), [286](#)
  - \_PLUQ, [361](#)
  - applyP, [304](#), [305](#), [363](#)
  - applyP\_block, [355](#)
  - Bruhat2EchelonPermutation, [343](#)
  - buildMatrix, [349](#)
  - CharPoly, [321](#), [322](#), [349](#), [367](#), [368](#)
  - Checker\_charpoly, [303](#)
  - Checker\_Det, [302](#)
  - Checker\_invert, [303](#)
  - Checker\_PLUQ, [302](#)
  - chooseField, [388](#)
  - chooseField< Givaro::ZRing< double > >, [389](#)
  - chooseField< Givaro::ZRing< float > >, [388](#)
  - chooseField< Givaro::ZRing< int32\_t > >, [388](#)
  - chooseField< Givaro::ZRing< int64\_t > >, [388](#)
  - ColRankProfileSubmatrix, [334](#), [372](#)
  - ColRankProfileSubmatrixIndices, [333](#), [372](#)
  - ColumnEchelonForm, [315](#), [316](#), [366](#)
  - ColumnRankProfile, [330](#), [331](#), [371](#)
  - composePermutationsLLL, [358](#)
  - composePermutationsLLM, [358](#)
  - composePermutationsMLM, [359](#)
  - CompressToBlockBiDiagonal, [342](#)
  - ComputeRPermutation, [344](#), [347](#)
  - cyclic\_shift\_col, [360](#), [363](#)
  - cyclic\_shift\_mathPerm, [359](#)
  - cyclic\_shift\_row, [359](#), [362](#)
  - cyclic\_shift\_row\_col, [359](#), [362](#)
  - Danilevski, [349](#)
  - Det, [326](#), [327](#), [350](#), [369](#), [370](#)
  - doApplyS, [355](#)
  - doApplyT, [356](#)
  - ExpandBlockBiDiagonalToBruhat, [343](#)
  - expandLCRE, [347](#)
  - failure, [375](#)
  - fflas\_const\_cast, [362](#), [375](#)
  - fgesv, [307](#), [308](#), [364](#)
  - fgetrs, [306](#), [363](#)
  - ForceCheck\_charpoly, [303](#)
  - ForceCheck\_Det, [303](#)
  - ForceCheck\_invert, [303](#)
  - ForceCheck\_PLUQ, [303](#)
  - fsytrf, [311](#), [312](#)
  - fsytrf\_BC\_Crout, [350](#)
  - fsytrf\_BC\_RL, [350](#)

- fsytrf\_LOW\_RPM\_BC\_Crout, 351
- fsytrf\_nonunit, 312, 352
- fsytrf\_RPM, 352
- fsytrf\_UP\_RPM, 351
- fsytrf\_UP\_RPM\_BC\_Crout, 351
- fsytrf\_UP\_RPM\_BC\_RL, 351
- ftssyr2k, 310
- ftstr, 310
- fttri, 309, 364
- fttrm, 309, 365
- getEchelonForm, 336
- getEchelonForm< FFLAS\_FIELD< FFLAS\_ELT >  
>, 373
- getEchelonTransform, 337
- getEchelonTransform< FFLAS\_FIELD< FFLAS\_ELT  
> >, 373
- getLTBruhatGen, 341
- getReducedEchelonForm, 338, 339
- getReducedEchelonForm< FFLAS\_FIELD<  
FFLAS\_ELT > >, 374
- getReducedEchelonTransform, 339
- getReducedEchelonTransform< FFLAS\_FIELD<  
FFLAS\_ELT > >, 374
- getTriangular, 335
- getTriangular< FFLAS\_FIELD< FFLAS\_ELT > >,  
372, 373
- getTridiagonal, 352
- Invert, 320, 367
- Invert2, 321, 367
- isOdd, 375
- IsSingular, 325, 369
- KrylovElim, 369
- LAPACKPerm2MathPerm, 303
- LeadingSubmatrixRankProfiles, 331
- LQUPtoInverseOfFullRankMinor, 345, 375
- LTBruhatGen, 340
- LTQSorter, 342
- LUdivine, 314, 353, 354, 365
- LUdivine\_gauss, 353, 366
- LUdivine\_small, 353, 365
- MathPerm2LAPACKPerm, 303
- MatrixApplyS, 355, 356
- MatrixApplyT, 357
- MatVecMinPoly, 324, 368
- MinPoly, 323, 368
- MonotonicApplyP, 305
- MonotonicCompress, 354
- MonotonicCompressCycles, 354
- MonotonicCompressMorePivots, 354
- MonotonicExpand, 355
- NonZeroRandomMatrix, 375, 376
- NullSpaceBasis, 328, 371
- pColumnEchelonForm, 316
- pColumnRankProfile, 331
- pDet, 326
- PermApplyS, 356
- PermApplyT, 358
- PLUQ, 313, 314, 361, 362, 365
- PLUQ\_basecaseCrout, 360
- PLUQ\_basecaseV2, 360
- PLUQ\_basecaseV3, 360
- PLUQtoEchelonPermutation, 340
- pPLUQ, 313
- pRank, 325
- pReducedColumnEchelonForm, 318
- pReducedRowEchelonForm, 319
- productBruhatxTS, 344, 347
- pRowEchelonForm, 317
- pRowRankProfile, 330
- pSolve, 328
- RandInt, 379
- RandomIndexSubset, 380
- RandomLTQSMatrixWithRankandQSorter, 388
- RandomLTQSRankProfileMatrix, 382
- RandomMatrix, 376, 377
- RandomMatrixWithDet, 387
- RandomMatrixWithRank, 379, 380
- RandomMatrixWithRankandRandomRPM, 385
- RandomMatrixWithRankandRPM, 382, 383
- RandomNullSpaceVector, 328, 345, 371
- RandomPermutation, 381
- RandomRankProfileMatrix, 381
- RandomSymmetricMatrix, 379
- RandomSymmetricMatrixWithRankandRandom-  
RPM, 386
- RandomSymmetricMatrixWithRankandRPM, 383,  
384
- RandomSymmetricRankProfileMatrix, 382
- RandomTriangularMatrix, 377, 378
- Rank, 324, 325, 369
- RankProfileFromLU, 331
- ReducedColumnEchelonForm, 317, 318, 367
- ReducedRowEchelonForm, 319, 366
- RowEchelonForm, 316, 317, 366
- RowRankProfile, 329, 330, 371
- RowRankProfileSubmatrix, 333, 372
- RowRankProfileSubmatrixIndices, 332, 372
- Solve, 327, 370
- solveLB, 346, 370
- solveLB2, 346, 370
- SpecRankProfile, 369
- swapval, 381
- threads\_fgemm, 361
- threads\_ftsm, 361
- TInverter, 344, 347
- trinv\_left, 309, 365
- ffpack-fgesv.C, 852
  - main, 853
- ffpack-solve.C, 853
  - main, 853
- ffpack.C, 728
  - applyP\_modular\_double, 734
  - ColRankProfileSubmatrix\_modular\_double, 746
  - ColRankProfileSubmatrixIndices\_modular\_double,  
746
  - ColumnEchelonForm\_modular\_double, 736

- ColumnEchelonForm\_modular\_float, [737](#)
- ColumnEchelonForm\_modular\_int32\_t, [738](#)
- ColumnRankProfile\_modular\_double, [745](#)
- composePermutationsLLL, [733](#)
- composePermutationsLLM, [733](#)
- composePermutationsMLM, [733](#)
- cyclic\_shift\_col\_modular\_double, [734](#)
- cyclic\_shift\_mathPerm, [733](#)
- cyclic\_shift\_row\_modular\_double, [733](#)
- Det\_modular\_double, [743](#)
- fgesv\_modular\_double, [735](#)
- fgesvin\_modular\_double, [735](#)
- fgetrsin\_modular\_double, [734](#)
- fgetrsv\_modular\_double, [734](#)
- fttrtri\_modular\_double, [735](#)
- fttrrm\_modular\_double, [736](#)
- getEchelonForm\_modular\_double, [747](#)
- getEchelonFormin\_modular\_double, [747](#)
- getEchelonTransform\_modular\_double, [747](#)
- getReducedEchelonForm\_modular\_double, [748](#)
- getReducedEchelonFormin\_modular\_double, [748](#)
- getReducedEchelonTransform\_modular\_double, [748](#)
- getTriangular\_modular\_double, [746](#)
- getTriangularin\_modular\_double, [747](#)
- Invert2\_modular\_double, [742](#)
- Invert\_modular\_double, [742](#)
- Invertin\_modular\_double, [742](#)
- IsSingular\_modular\_double, [743](#)
- KrylovElim\_modular\_double, [742](#)
- LAPACKPerm2MathPerm, [732](#)
- LeadingSubmatrixRankProfiles, [745](#)
- LUdivine\_modular\_double, [736](#)
- MathPerm2LAPACKPerm, [732](#)
- MatrixApplyS\_modular\_double, [732](#)
- MatrixApplyT\_modular\_double, [732](#)
- NullSpaceBasis\_modular\_double, [744](#)
- pColumnEchelonForm\_modular\_double, [739](#)
- pColumnEchelonForm\_modular\_float, [740](#)
- pColumnEchelonForm\_modular\_int32\_t, [741](#)
- PermApplyS\_double, [732](#)
- PermApplyT\_double, [733](#)
- PLUQ\_modular\_double, [736](#)
- PLUQtoEchelonPermutation, [748](#)
- pReducedColumnEchelonForm\_modular\_double, [740](#)
- pReducedColumnEchelonForm\_modular\_float, [740](#)
- pReducedColumnEchelonForm\_modular\_int32\_t, [741](#)
- pReducedRowEchelonForm\_modular\_double, [740](#)
- pReducedRowEchelonForm\_modular\_float, [741](#)
- pReducedRowEchelonForm\_modular\_int32\_t, [742](#)
- pRowEchelonForm\_modular\_double, [739](#)
- pRowEchelonForm\_modular\_float, [740](#)
- pRowEchelonForm\_modular\_int32\_t, [741](#)
- RandomNullSpaceVector\_modular\_double, [744](#)
- Rank\_modular\_double, [743](#)
- RankProfileFromLU, [745](#)
- ReducedColumnEchelonForm\_modular\_double, [737](#)
- ReducedColumnEchelonForm\_modular\_float, [738](#)
- ReducedColumnEchelonForm\_modular\_int32\_t, [739](#)
- ReducedRowEchelonForm\_modular\_double, [737](#)
- ReducedRowEchelonForm\_modular\_float, [738](#)
- ReducedRowEchelonForm\_modular\_int32\_t, [739](#)
- RowEchelonForm\_modular\_double, [736](#)
- RowEchelonForm\_modular\_float, [737](#)
- RowEchelonForm\_modular\_int32\_t, [738](#)
- RowRankProfile\_modular\_double, [745](#)
- RowRankProfileSubmatrix\_modular\_double, [746](#)
- RowRankProfileSubmatrixIndices\_modular\_double, [745](#)
- Solve\_modular\_double, [743](#)
- solveLB2\_modular\_double, [744](#)
- solveLB\_modular\_double, [744](#)
- SpecRankProfile\_modular\_double, [743](#)
- trinv\_left\_modular\_double, [735](#)
- ffpack.doxy, [659](#)
- ffpack.h, [659](#)
  - \_\_FFLASFFPACK\_FTRSSYR2K\_THRESHOLD, [667](#)
  - \_\_FFLASFFPACK\_FTRSTR\_THRESHOLD, [667](#)
- ffpack.inl, [668](#)
  - \_\_FFLASFFPACK\_ffpack\_INL, [669](#)
- FFPACK::Protected, [389](#)
  - ArithProg, [393](#)
  - CompressRows, [394](#)
  - CompressRowsQA, [395](#)
  - CompressRowsQK, [395](#)
  - Danilevski, [393](#)
  - DeCompressRows, [395](#)
  - DeCompressRowsQA, [396](#)
  - DeCompressRowsQK, [395](#)
  - fgemv\_kgf, [392](#)
  - GaussJordan, [391](#)
  - Hybrid\_KGF\_LUK\_MinPoly, [394](#)
  - KellerGehrig, [391](#)
  - KGFast, [392](#)
  - KGFast\_generalized, [392](#)
  - LUdivine\_construct, [390](#), [396](#)
  - LUKrylov, [392](#)
  - LUKrylov\_KGFast, [393](#)
  - MatVecMinPoly, [393](#)
  - newD, [394](#)
  - RandomKrylovPrecond, [393](#)
  - updateD, [394](#)
- ffpack\_bruhatgen.inl, [669](#)
  - \_\_FFLASFFPACK\_ffpack\_bruhatgen\_inl, [670](#)
- ffpack\_c.h, [749](#)
  - applyP\_modular\_double, [755](#)
  - ColRankProfileSubmatrix\_modular\_double, [766](#)
  - ColRankProfileSubmatrixIndices\_modular\_double, [765](#)
  - ColumnEchelonForm\_modular\_double, [758](#)

- ColumnEchelonForm\_modular\_float, 759
- ColumnEchelonForm\_modular\_int32\_t, 759
- ColumnRankProfile\_modular\_double, 764
- composePermutationsLLL, 755
- composePermutationsLLM, 754
- composePermutationsMLM, 755
- cyclic\_shift\_col\_modular\_double, 755
- cyclic\_shift\_mathPerm, 755
- cyclic\_shift\_row\_modular\_double, 755
- Det\_modular\_double, 763
- FFLAS\_C\_DIAG, 752
- FFLAS\_C\_ORDER, 752
- FFLAS\_C\_SIDE, 752
- FFLAS\_C\_TRANSPOSE, 752
- FFLAS\_C\_UPLO, 752
- FflasColMajor, 752
- FflasLeft, 753
- FflasLower, 752
- FflasNonUnit, 752
- FflasNoTrans, 752
- FflasRight, 753
- FflasRowMajor, 752
- FflasTrans, 752
- FflasUnit, 752
- FflasUpper, 752
- FFPACK\_C\_CHARPOLY\_TAG, 753
- FFPACK\_C\_LU\_TAG, 753
- FFPACK\_C\_MINPOLY\_TAG, 753
- FFPACK\_COMPILED, 752
- FfpackArithProg, 753
- FfpackDanilevski, 753
- FfpackDense, 753
- FfpackHybrid, 753
- FfpackKG, 753
- FfpackKGF, 753
- FfpackKGFast, 753
- FfpackKGFastG, 753
- FfpackLUK, 753
- FfpackSingular, 753
- FfpackSlabRecursive, 753
- FfpackTileRecursive, 753
- fgesv\_modular\_double, 756
- fgesvin\_modular\_double, 756
- fgetrs\_modular\_double, 756
- fgetrsin\_modular\_double, 756
- fttrtri\_modular\_double, 757
- fttrrm\_modular\_double, 757
- getEchelonForm\_modular\_double, 766
- getEchelonFormin\_modular\_double, 767
- getEchelonTransform\_modular\_double, 767
- getReducedEchelonForm\_modular\_double, 767
- getReducedEchelonFormin\_modular\_double, 768
- getReducedEchelonTransform\_modular\_double, 768
- getTriangular\_modular\_double, 766
- getTriangularin\_modular\_double, 766
- Invert2\_modular\_double, 762
- Invert\_modular\_double, 762
- Invertin\_modular\_double, 762
- IsSingular\_modular\_double, 763
- KrylovElim\_modular\_double, 762
- LAPACKPerm2MathPerm, 753
- LeadingSubmatrixRankProfiles, 765
- LUdivine\_gauss\_modular\_double, 758
- LUdivine\_modular\_double, 757
- LUdivine\_small\_modular\_double, 758
- MathPerm2LAPACKPerm, 753
- MatrixApplyS\_modular\_double, 754
- MatrixApplyT\_modular\_double, 754
- NullSpaceBasis\_modular\_double, 764
- PermApplyS\_double, 754
- PermApplyT\_double, 754
- PLUQ\_modular\_double, 757
- PLUQtoEchelonPermutation, 768
- RandomNullSpaceVector\_modular\_double, 764
- Rank\_modular\_double, 763
- RankProfileFromLU, 765
- ReducedColumnEchelonForm\_modular\_double, 760
- ReducedColumnEchelonForm\_modular\_float, 760
- ReducedColumnEchelonForm\_modular\_int32\_t, 761
- ReducedRowEchelonForm2\_modular\_double, 761
- ReducedRowEchelonForm\_modular\_double, 760
- ReducedRowEchelonForm\_modular\_float, 760
- ReducedRowEchelonForm\_modular\_int32\_t, 761
- REF\_modular\_double, 761
- RowEchelonForm\_modular\_double, 758
- RowEchelonForm\_modular\_float, 759
- RowEchelonForm\_modular\_int32\_t, 759
- RowRankProfile\_modular\_double, 764
- RowRankProfileSubmatrix\_modular\_double, 765
- RowRankProfileSubmatrixIndices\_modular\_double, 765
- Solve\_modular\_double, 763
- solveLB2\_modular\_double, 764
- solveLB\_modular\_double, 763
- SpecRankProfile\_modular\_double, 762
- trinv\_left\_modular\_double, 757
- FFPACK\_C\_CHARPOLY\_TAG
  - ffpack\_c.h, 753
- FFPACK\_C\_LU\_TAG
  - ffpack\_c.h, 753
- FFPACK\_C\_MINPOLY\_TAG
  - ffpack\_c.h, 753
- ffpack\_charpoly.inl, 670
  - \_\_FFLASFFPACK\_charpoly\_INL, 671
- ffpack\_charpoly\_danilevski.inl, 671
  - \_\_FFLASFFPACK\_ffpack\_charpoly\_danilveski\_INL, 671
- ffpack\_charpoly\_kgfast.inl, 671
  - \_\_FFLASFFPACK\_ffpack\_charpoly\_kgfast\_INL, 672
- ffpack\_charpoly\_kgfastgeneralized.inl, 672
  - \_\_FFLASFFPACK\_ffpack\_charpoly\_kgfastgeneralized\_INL, 672

ffpack\_charpoly\_kglu.inl, 672  
     \_\_FFLASFFPACK\_ffpack\_charpoly\_kglu\_INL, 673  
 ffpack\_charpoly\_mp.inl, 673  
     \_\_FFPACK\_charpoly\_mp\_INL, 673  
 FFPACK\_COMPILED  
     ffpack\_c.h, 752  
 ffpack\_det\_mp.inl, 673  
     \_\_FFPACK\_det\_mp\_INL, 674  
 ffpack\_echelonforms.inl, 674  
     \_\_FFLASFFPACK\_GAUSSJORDAN\_BASECASE, 675  
     \_\_FFLASFFPACK\_ffpack\_echelon\_forms\_INL, 675  
 ffpack\_fgesv.inl, 675  
     \_\_FFLASFFPACK\_ffpack\_fgesv\_INL, 676  
 ffpack\_fgetrs.inl, 676  
     \_\_FFLASFFPACK\_ffpack\_fgetrs\_INL, 676  
 ffpack\_frobenius.inl, 676  
 ffpack\_fsytrf.inl, 677  
     \_\_FFLASFFPACK\_ffpack\_fsytrf\_INL, 678  
 ffpack\_ftrssyr2k.inl, 678  
     \_\_FFLASFFPACK\_ffpack\_ftrssyr2k\_INL, 679  
 ffpack\_ftrstr.inl, 679  
     \_\_FFLASFFPACK\_ffpack\_ftrstr\_INL, 679  
 ffpack\_ftrtr.inl, 679  
     \_\_FFLASFFPACK\_ffpack\_ftrtr\_INL, 680  
     ENABLE\_ALL\_CHECKINGS, 680  
 ffpack\_inst.C, 768  
     \_\_FFPACK\_INST\_C, 769  
     FFLAS\_COMPILED, 769  
     FFLAS\_ELT, 769  
     FFLAS\_FIELD, 769  
     INST\_OR\_DECL, 769  
 ffpack\_inst.h, 769  
     FFLAS\_COMPILED, 770  
     FFLAS\_ELT, 770  
     FFLAS\_FIELD, 770  
     INST\_OR\_DECL, 770  
 ffpack\_inst\_implem.inl, 771  
 ffpack\_invert.inl, 680  
     \_\_FFLASFFPACK\_ffpack\_invert\_INL, 680  
 ffpack\_krylovelim.inl, 681  
     \_\_FFLASFFPACK\_ffpack\_krylovelim\_INL, 681  
 ffpack\_ludivine.inl, 681  
     \_\_FFLASFFPACK\_ffpack\_ludivine\_INL, 681  
 ffpack\_ludivine\_mp.inl, 682  
     \_\_FFPACK\_ludivine\_mp\_INL, 682  
 ffpack\_minpoly.inl, 682  
     \_\_FFLASFFPACK\_ffpack\_minpoly\_INL, 683  
 ffpack\_permutation.inl, 683  
     \_\_FFLASFFPACK\_ffpack\_permutation\_INL, 685  
     FFLASFFPACK\_PERM\_BKSIZE, 685  
 ffpack\_pluq.inl, 685  
     \_\_FFLASFFPACK\_ffpack\_pluq\_INL, 686  
     CROUT, 686  
 ffpack\_pluq\_mp.inl, 686  
     \_\_FFPACK\_pluq\_mp\_INL, 686  
 ffpack\_ppluq.inl, 686  
     \_\_FFLASFFPACK\_ffpack\_ppluq\_INL, 687  
     \_\_FFLAS\_\_TRSM\_READONLY, 687  
     PBASECASE\_K, 687  
 ffpack\_rankprofiles.inl, 687  
     \_\_FFLASFFPACK\_ffpack\_rank\_profiles\_INL, 688  
 FfpackArithProg  
     ffpack\_c.h, 753  
 FfpackDanilevski  
     ffpack\_c.h, 753  
 FfpackDense  
     ffpack\_c.h, 753  
 FfpackHybrid  
     ffpack\_c.h, 753  
 FfpackKG  
     ffpack\_c.h, 753  
 FfpackKGF  
     ffpack\_c.h, 753  
 FfpackKGFast  
     ffpack\_c.h, 753  
 FfpackKGFastG  
     ffpack\_c.h, 753  
 FfpackLUK  
     ffpack\_c.h, 753  
 FfpackSingular  
     ffpack\_c.h, 753  
 FfpackSlabRecursive  
     ffpack\_c.h, 753  
 FfpackTileRecursive  
     ffpack\_c.h, 753  
 fgemm  
     FFLAS, 88–90, 92–97, 144, 176–178  
 fgemm\_3\_modular\_double  
     fflas\_c.h, 706  
     fflas\_lvl3.C, 727  
 fgemm\_classical.inl, 579  
 fgemm\_classical\_mp.inl, 579  
     \_\_FFPACK\_fgemm\_classical\_INL, 581  
 fgemm\_convert  
     FFLAS::Protected, 212  
 fgemm\_winograd.inl, 581  
     \_\_FFLASFFPACK\_fflas\_fflas\_fgemm\_winograd\_INL, 583  
     NEWWINO, 583  
 fgemv  
     FFLAS, 97–102, 173, 184  
 fgemv\_2\_modular\_double  
     fflas\_c.h, 705  
     fflas\_lvl2.C, 725  
 fgemv\_convert  
     FFLAS::Protected, 217  
 fgemv\_kgf  
     FFPACK::Protected, 392  
 fger  
     FFLAS, 103–106, 173  
 fger\_2\_modular\_double  
     fflas\_c.h, 705  
     fflas\_lvl2.C, 726  
 fger\_convert

- FFLAS::Protected, 218
- fgesv
  - FFPACK, 307, 308, 364
- fgesv\_modular\_double
  - ffpack.C, 735
  - ffpack\_c.h, 756
- fgesvin\_modular\_double
  - ffpack.C, 735
  - ffpack\_c.h, 756
- fgetrs
  - FFPACK, 306, 363
- fgetrs\_modular\_double
  - ffpack\_c.h, 756
- fgetrsin\_modular\_double
  - ffpack.C, 734
  - ffpack\_c.h, 756
- fgetrsv\_modular\_double
  - ffpack.C, 734
- fidentity
  - FFLAS, 136, 137, 166
- fidentity\_2\_modular\_double
  - fflas\_c.h, 702
  - fflas\_lvl2.C, 723
- Field
  - Bench< Elt >, 400
  - benchmark-fgemm-rns.C, 532
  - benchmark-pluq.C, 545
  - FieldSimd< \_Field >, 426
  - Test< Elt >, 506
  - test-compressQ.C, 800
- field
  - associatedDelayedField< Field >, 399
- field-traits.h, 688
- field.dox, 691
- FieldMax
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, 455
- FieldMin
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, 455
- FieldSimd
  - FieldSimd< \_Field >, 426
- FieldSimd< \_Field >, 425
  - add, 427
  - add\_r, 427
  - addin, 427
  - addin\_r, 427
  - alignment, 430
  - axpy, 429
  - axpy\_r, 430
  - axpyin, 429
  - axpyin\_r, 430
  - Element, 426
  - Field, 426
  - FieldSimd, 426
  - init, 426, 427
  - maxpy, 430
  - maxpyin, 430
  - mod, 428
  - mul, 428, 429
  - mul\_r, 429
  - mulin, 429
  - operator=, 426
  - scalar\_t, 426
  - simd, 426
  - sub, 427, 428
  - sub\_r, 428
  - subin, 428
  - subin\_r, 428
  - vect\_size, 430
  - vect\_t, 426
  - zero, 428
- FieldTraits< Field >, 431
  - balanced, 431
  - category, 431
- fill\_value
  - benchmark-fgemv.C, 536
- findArgument
  - args-parser.h, 786
- finit
  - FFLAS, 108, 110, 129, 137, 158, 167
- finit\_rns
  - FFLAS, 155, 156
- finit\_trans\_rns
  - FFLAS, 155
- first\_component
  - Compose< H1, H2 >, 413
- firstBlockSize
  - ForStrategy1D< blocksize\_t, Cut, Param >, 434
- fiszero
  - FFLAS, 131, 136, 160, 166
- fiszero\_1\_modular\_double
  - fflas\_c.h, 699
  - fflas\_lvl1.C, 719
- fiszero\_2\_modular\_double
  - fflas\_c.h, 702
  - fflas\_lvl2.C, 723
- Fixed, 431
- FixedPrecIntTag, 431
- flimits.h, 792
  - in\_range, 792, 793
- FLOAT\_MOD
  - fflas\_simd.h, 617
- Floats
  - benchmark-dgemm.C, 526
- fmadd
  - ScalFunctions< Element >, 493
- fmaddin
  - ScalFunctions< Element >, 493
- fmove
  - FFLAS, 139, 170
- fmove\_2\_modular\_double
  - fflas\_c.h, 704
  - fflas\_lvl2.C, 724
- fmsub
  - ScalFunctions< Element >, 493

- fmsubin
  - ScalFunctions< Element >, [493](#)
- fneg
  - FFLAS, [130](#), [138](#), [159](#), [168](#)
- fneg\_1\_modular\_double
  - fflas\_c.h, [699](#)
  - fflas\_lvl1.C, [718](#)
- fneg\_2\_modular\_double
  - fflas\_c.h, [703](#)
  - fflas\_lvl2.C, [724](#)
- fnegin
  - FFLAS, [130](#), [138](#), [159](#), [168](#)
- fnegin\_1\_modular\_double
  - fflas\_c.h, [699](#)
  - fflas\_lvl1.C, [718](#)
- fnegin\_2\_modular\_double
  - fflas\_c.h, [703](#)
  - fflas\_lvl2.C, [723](#)
- fnmadd
  - ScalFunctions< Element >, [493](#)
- fnmaddin
  - ScalFunctions< Element >, [494](#)
- FOR1D
  - parallel.h, [779](#)
- FOR2D
  - parallel.h, [779](#)
- FORBLOCK1D
  - parallel.h, [778](#)
- FORBLOCK2D
  - parallel.h, [779](#)
- ForceCheck\_charpoly
  - FFPACK, [303](#)
- ForceCheck\_Det
  - FFPACK, [303](#)
- ForceCheck\_fgemm
  - FFLAS, [72](#)
- ForceCheck\_ftsm
  - FFLAS, [72](#)
- ForceCheck\_invert
  - FFPACK, [303](#)
- ForceCheck\_PLUQ
  - FFPACK, [303](#)
- ForStrategy1D
  - ForStrategy1D< blocksize\_t, Cut, Param >, [432](#)
- ForStrategy1D< blocksize\_t, Cut, Param >, [432](#)
  - begin, [433](#)
  - blockindex, [433](#)
  - build, [432](#)
  - changeBS, [434](#)
  - current, [433](#)
  - end, [433](#)
  - firstBlockSize, [434](#)
  - ForStrategy1D, [432](#)
  - ibeg, [433](#)
  - iend, [433](#)
  - initialize, [433](#)
  - isTerminated, [433](#)
  - lastBlockSize, [434](#)
  - numBlock, [434](#)
  - numblocks, [433](#)
  - operator++, [433](#)
- ForStrategy2D
  - ForStrategy2D< blocksize\_t, Cut, Param >, [435](#)
- ForStrategy2D< blocksize\_t, Cut, Param >, [434](#)
  - \_ibeg, [436](#)
  - \_iend, [436](#)
  - \_jbeg, [437](#)
  - \_jend, [437](#)
  - blockindex, [436](#)
  - BLOCKS, [438](#)
  - changeCBS, [437](#)
  - changeRBS, [437](#)
  - colblockindex, [436](#)
  - colBlockSize, [437](#)
  - colnumblocks, [436](#)
  - current, [437](#)
  - ForStrategy2D, [435](#)
  - ibegin, [435](#)
  - iend, [435](#)
  - initialize, [435](#)
  - isTerminated, [435](#)
  - jbegin, [435](#)
  - jend, [435](#)
  - lastCBS, [437](#)
  - lastRBS, [437](#)
  - numColBlock, [438](#)
  - numRowBlock, [437](#)
  - operator<<, [436](#)
  - operator++, [436](#)
  - rowblockindex, [436](#)
  - rowBlockSize, [437](#)
  - rownumblocks, [436](#)
- frand
  - FFLAS, [131](#), [135](#)
- freduce
  - FFLAS, [107–111](#), [157](#), [167](#)
  - FFLAS::details, [201](#), [202](#)
- freduce\_1\_modular\_double
  - fflas\_c.h, [698](#)
  - fflas\_lvl1.C, [718](#)
- freduce\_2\_modular\_double
  - fflas\_c.h, [703](#)
  - fflas\_lvl2.C, [723](#)
- freduce\_constoverride
  - FFLAS, [108](#), [110](#)
- freducein\_1\_modular\_double
  - fflas\_c.h, [698](#)
  - fflas\_lvl1.C, [718](#)
- freducein\_2\_modular\_double
  - fflas\_c.h, [702](#)
  - fflas\_lvl2.C, [723](#)
- freivalds
  - FFLAS, [111](#)
- fscal
  - FFLAS, [112–116](#), [161](#), [169](#)
  - FFLAS::details, [202](#), [203](#)



fscal\_1\_modular\_double  
     fflas\_c.h, 700  
     fflas\_lvl1.C, 719  
 fscal\_2\_modular\_double  
     fflas\_c.h, 703  
     fflas\_lvl2.C, 724  
 fscaln  
     FFLAS, 111, 113–116, 161, 169  
     FFLAS::details, 202, 203  
 fscaln\_1\_modular\_double  
     fflas\_c.h, 700  
     fflas\_lvl1.C, 719  
 fscaln\_2\_modular\_double  
     fflas\_c.h, 703  
     fflas\_lvl2.C, 724  
 fspmm  
     FFLAS, 145  
     FFLAS::sparse\_details, 229–231  
     FFLAS::sparse\_details\_impl, 245, 252, 253, 258, 259, 263, 272  
 fspmm\_dispatch  
     FFLAS::sparse\_details, 228, 229  
 fspmm\_mone  
     FFLAS::sparse\_details\_impl, 246, 254, 264  
 fspmm\_mone\_simd\_aligned  
     FFLAS::sparse\_details\_impl, 247, 254, 265  
 fspmm\_mone\_simd\_unaligned  
     FFLAS::sparse\_details\_impl, 247, 255, 265  
 fspmm\_one  
     FFLAS::sparse\_details\_impl, 246, 253, 264  
 fspmm\_one\_simd\_aligned  
     FFLAS::sparse\_details\_impl, 247, 254, 265  
 fspmm\_one\_simd\_unaligned  
     FFLAS::sparse\_details\_impl, 247, 254, 265  
 fspmm\_simd\_aligned  
     FFLAS::sparse\_details\_impl, 246, 253  
 fspmm\_simd\_unaligned  
     FFLAS::sparse\_details\_impl, 246, 253  
 fspmv  
     FFLAS, 145, 152  
     FFLAS::sparse\_details, 226–228, 236  
     FFLAS::sparse\_details\_impl, 248, 255, 259, 266, 268, 269, 273, 275  
 fspmv\_dispatch  
     FFLAS::sparse\_details, 226  
 fspmv\_mone  
     FFLAS::sparse\_details\_impl, 248, 249, 256, 266, 267, 270, 276, 277  
 fspmv\_mone\_simd  
     FFLAS::sparse\_details\_impl, 270, 276  
 fspmv\_one  
     FFLAS::sparse\_details\_impl, 248, 249, 255, 256, 266, 267, 269, 270, 276  
 fspmv\_one\_simd  
     FFLAS::sparse\_details\_impl, 270, 276  
 fspmv\_simd  
     FFLAS::sparse\_details\_impl, 269, 275  
 fsquare  
     FFLAS, 91, 92, 178  
 fsquare\_3\_modular\_double  
     fflas\_c.h, 707  
     fflas\_lvl3.C, 728  
 fsquareCommon  
     FFLAS::Protected, 215  
 fsub  
     FFLAS, 78, 80, 164, 171  
 fsub\_1\_modular\_double  
     fflas\_c.h, 701  
     fflas\_lvl1.C, 720  
 fsub\_2\_modular\_double  
     fflas\_c.h, 704  
     fflas\_lvl2.C, 725  
 fsubin  
     FFLAS, 78, 81, 172  
 fsubin\_1\_modular\_double  
     fflas\_c.h, 701  
     fflas\_lvl1.C, 721  
 fsubin\_2\_modular\_double  
     fflas\_c.h, 704  
     fflas\_lvl2.C, 725  
 fswap  
     FFLAS, 133, 163  
 fswap\_1\_modular\_double  
     fflas\_c.h, 700  
     fflas\_lvl1.C, 720  
 fsyr2k  
     FFLAS, 116  
 fsyrk  
     FFLAS, 117–124  
 fsyrk.C, 519  
     CUBE, 519  
     GFOPS, 519  
     main, 519  
 fsyrk\_convert  
     FFLAS::Protected, 218  
 fsyrk\_strassen  
     FFLAS, 124, 142  
 fsytrf  
     FFPACK, 311, 312  
 fsytrf.C, 519  
     CUBE, 520  
     GFOPS, 520  
     main, 520  
 fsytrf\_BC\_Crout  
     FFPACK, 350  
 fsytrf\_BC\_RL  
     FFPACK, 350  
 fsytrf\_LOW\_RPM\_BC\_Crout  
     FFPACK, 351  
 fsytrf\_nonunit  
     FFPACK, 312, 352  
 fsytrf\_RPM  
     FFPACK, 352  
 fsytrf\_UP\_RPM  
     FFPACK, 351  
 fsytrf\_UP\_RPM\_BC\_Crout



- FFPACK, 351
- fsytrf\_UP\_RPM\_BC\_RL
  - FFPACK, 351
- ftranspose
  - FFLAS, 154
- ftmm
  - FFLAS, 124, 125, 175
- ftmm\_3\_modular\_double
  - fflas\_c.h, 706
  - fflas\_lvl3.C, 727
- ftmmLeftLowerNoTransNonUnit< Element >, 438
- ftmmLeftLowerNoTransUnit< Element >, 438
- ftmmLeftLowerTransNonUnit< Element >, 438
- ftmmLeftLowerTransUnit< Element >, 438
- ftmmLeftUpperNoTransNonUnit< Element >, 438
- ftmmLeftUpperNoTransUnit< Element >, 439
- ftmmLeftUpperTransNonUnit< Element >, 439
- ftmmLeftUpperTransUnit< Element >, 439
- ftmmRightLowerNoTransNonUnit< Element >, 439
- ftmmRightLowerNoTransUnit< Element >, 439
- ftmmRightLowerTransNonUnit< Element >, 439
- ftmmRightLowerTransUnit< Element >, 439
- ftmmRightUpperNoTransNonUnit< Element >, 439
- ftmmRightUpperNoTransUnit< Element >, 440
- ftmmRightUpperTransNonUnit< Element >, 440
- ftmmRightUpperTransUnit< Element >, 440
- ftmv
  - FFLAS, 140
- ftsm
  - FFLAS, 126, 127, 141, 144, 145, 175
- ftsm\_3\_modular\_double
  - fflas\_c.h, 706
  - fflas\_lvl3.C, 727
- ftsmLeftLowerNoTransNonUnit< Element >, 440
- ftsmLeftLowerNoTransUnit< Element >, 440
- ftsmLeftLowerTransNonUnit< Element >, 440
- ftsmLeftLowerTransUnit< Element >, 440
- ftsmLeftUpperNoTransNonUnit< Element >, 440
- ftsmLeftUpperNoTransUnit< Element >, 441
- ftsmLeftUpperTransNonUnit< Element >, 441
- ftsmLeftUpperTransUnit< Element >, 441
- ftsmRightLowerNoTransNonUnit< Element >, 441
- ftsmRightLowerNoTransUnit< Element >, 441
- ftsmRightLowerTransNonUnit< Element >, 442
- ftsmRightLowerTransUnit< Element >, 442
- ftsmRightUpperNoTransNonUnit< Element >, 442
- ftsmRightUpperNoTransUnit< Element >, 442
- ftsmRightUpperTransNonUnit< Element >, 442
- ftsmRightUpperTransUnit< Element >, 442
- ftssyr2k
  - FFPACK, 310
- ftstr
  - FFPACK, 310
- ftsv
  - FFLAS, 128, 174
- ftsv\_2\_modular\_double
  - fflas\_c.h, 705
  - fflas\_lvl2.C, 726
- fttri
  - FFPACK, 309, 364
- fttri.C, 520
  - CUBE, 520
  - GFOPS, 520
  - main, 521
- fttri\_modular\_double
  - ffpack.C, 735
  - ffpack\_c.h, 757
- fttrm
  - FFPACK, 309, 365
- fttrm\_modular\_double
  - ffpack.C, 736
  - ffpack\_c.h, 757
- fzero
  - FFLAS, 130, 133, 135, 159, 165
- fzero\_1\_modular\_double
  - fflas\_c.h, 699
  - fflas\_lvl1.C, 718
- fzero\_2\_modular\_double
  - fflas\_c.h, 702
  - fflas\_lvl2.C, 722
- GaussJordan
  - FFPACK::Protected, 391
- GCC\_VERSION
  - fflas-ffpack-config.h, 568
- gebp
  - FFLAS::details, 205
- genData
  - benchmark-fgemv.C, 536
- GenericTag, 442, 443
- genInputs
  - ScalFunctions< Element >, 491
- genInputsWithZero
  - ScalFunctions< Element >, 491
- getArgumentValue
  - FFLAS, 185
- getDataType
  - FFLAS, 151, 152
- getEchelonForm
  - FFPACK, 336
- getEchelonForm< FFLAS\_FIELD< FFLAS\_ELT > >
  - FFPACK, 373
- getEchelonForm\_modular\_double
  - ffpack.C, 747
  - ffpack\_c.h, 766
- getEchelonFormin\_modular\_double
  - ffpack.C, 747
  - ffpack\_c.h, 767
- getEchelonTransform
  - FFPACK, 337
- getEchelonTransform< FFLAS\_FIELD< FFLAS\_ELT > >
  - FFPACK, 373
- getEchelonTransform\_modular\_double
  - ffpack.C, 747
  - ffpack\_c.h, 767
- getListArgs

- args-parser.h, 786
- getLTBruhatGen
  - FFPACK, 341
- getReducedEchelonForm
  - FFPACK, 338, 339
- getReducedEchelonForm< FFLAS\_FIELD< FFLAS\_ELT  
> >
  - FFPACK, 374
- getReducedEchelonForm\_modular\_double
  - ffpack.C, 748
  - ffpack\_c.h, 767
- getReducedEchelonFormin\_modular\_double
  - ffpack.C, 748
  - ffpack\_c.h, 768
- getReducedEchelonTransform
  - FFPACK, 339
- getReducedEchelonTransform< FFLAS\_FIELD< FFLAS\_ELT > >
  - FFPACK, 374
- getReducedEchelonTransform\_modular\_double
  - ffpack.C, 748
  - ffpack\_c.h, 768
- getSeed
  - FFLAS, 189
- getStat
  - FFLAS, 154
- getStatus
  - TestOneMethod< Simd >, 510
- getTestName
  - TestOneMethod< Simd >, 510
- getTLBSize
  - FFLAS, 189
- getTriangular
  - FFPACK, 335
- getTriangular< FFLAS\_FIELD< FFLAS\_ELT > >
  - FFPACK, 372, 373
- getTriangular\_modular\_double
  - ffpack.C, 746
  - ffpack\_c.h, 766
- getTriangularin\_modular\_double
  - ffpack.C, 747
  - ffpack\_c.h, 766
- getTridiagonal
  - FFPACK, 352
- gf2ModularBalanced
  - regression-check.C, 798
- GFOPS
  - charpoly.C, 518
  - fsyrk.C, 519
  - fsytrf.C, 520
  - fttrtri.C, 520
  - pluq.C, 521
  - winograd.C, 522
- Givaro, 396
- GRAIN
  - benchmark-fgemm-rns.C, 532
- Grain, 443
- greater
  - ScalFunctions< Element >, 494
- greater\_eq
  - ScalFunctions< Element >, 494
- has\_equal
  - FFLAS, 73
- has\_minus
  - FFLAS, 73
- has\_minus\_eq
  - FFLAS, 73
- has\_minus\_eq\_impl< C >, 443
  - value, 443
- has\_minus\_impl< C >, 443
  - value, 443
- has\_mul
  - FFLAS, 73
- has\_mul\_eq
  - FFLAS, 74
- has\_mul\_eq\_impl< C >, 443
  - value, 444
- has\_mul\_impl< C >, 444
  - value, 444
- has\_operation< T >, 444
  - value, 444
- has\_plus
  - FFLAS, 73
- has\_plus\_eq
  - FFLAS, 73
- has\_plus\_eq\_impl< C >, 444
  - value, 445
- has\_plus\_impl< C >, 445
  - value, 445
- HAVE\_BLAS
  - config.h, 561
- HAVE\_CBLAS
  - config.h, 561
- HAVE\_CXX11
  - config.h, 561
- HAVE\_DLFCN\_H
  - config.h, 562
- HAVE\_FLOAT\_H
  - config.h, 562
- HAVE\_INT128
  - config.h, 562
- HAVE\_INTTYPES\_H
  - config.h, 562
- HAVE\_LAPACK
  - config.h, 562
- HAVE\_LIMITS\_H
  - config.h, 562
- HAVE\_LITTLE\_ENDIAN
  - config.h, 562
- HAVE\_PTHREAD\_H
  - config.h, 562
- HAVE\_STDDEF\_H
  - config.h, 562
- HAVE\_STDINT\_H
  - config.h, 562
- HAVE\_STDIO\_H

- config.h, [562](#)
- HAVE\_STDLIB\_H
  - config.h, [562](#)
- HAVE\_STRING\_H
  - config.h, [562](#)
- HAVE\_STRINGS\_H
  - config.h, [562](#)
- HAVE\_SYS\_STAT\_H
  - config.h, [562](#)
- HAVE\_SYS\_TIME\_H
  - config.h, [563](#)
- HAVE\_SYS\_TYPES\_H
  - config.h, [563](#)
- HAVE\_UNISTD\_H
  - config.h, [563](#)
- HelperFlag, [445](#)
  - aut, [445](#)
  - coo, [445](#)
  - csr, [445](#)
  - ell, [445](#)
  - none, [445](#)
  - pm1, [446](#)
- HelperMod< Field, ElementTraits >, [446](#)
- helpString
  - Argument, [398](#)
- HYB\_ZO
  - FFLAS, [76](#)
- hyb\_zo.h, [649](#)
- hyb\_zo\_pspmm.inl, [649](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_pspmm\_INL, [650](#)
- hyb\_zo\_pspmv.inl, [650](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_pspmv\_INL, [650](#)
- hyb\_zo\_spm.inl, [650](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_spm\_INL, [651](#)
- hyb\_zo\_spmv.inl, [651](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_spmv\_INL, [651](#)
- hyb\_zo\_utils.inl, [651](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_utils\_INL, [652](#)
- Hybrid, [446](#)
- Hybrid\_KGF\_LUK\_MinPoly
  - FFPACK::Protected, [394](#)
- ibeg
  - ForStrategy1D< blocksize\_t, Cut, Param >, [433](#)
- ibegin
  - ForStrategy2D< blocksize\_t, Cut, Param >, [435](#)
- idamax\_
  - config-blas.h, [557](#)
- iend
  - ForStrategy1D< blocksize\_t, Cut, Param >, [433](#)
  - ForStrategy2D< blocksize\_t, Cut, Param >, [435](#)
- igebb11
  - FFLAS::details, [204](#)
- 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, [221](#)
- igemm.doxy, [603](#)
- igemm.h, [603](#)
- igemm.inl, [603](#)
  - \_\_FFLASFFPACK\_fflas\_igemm\_igemm\_INL, [604](#)
- igemm\_
  - FFLAS, [128](#)
- igemm\_colmajor
  - FFLAS::Protected, [221](#)
- igemm\_kernels.h, [604](#)
- igemm\_kernels.inl, [605](#)
  - \_\_FFLASFFPACK\_fflas\_igemm\_igemm\_kernels\_INL, [605](#)
- igemm\_tools.h, [605](#)
- igemm\_tools.inl, [606](#)
  - \_\_FFLASFFPACK\_fflas\_igemm\_igemm\_tools\_INL, [606](#)
- inl\_range
  - flimits.h, [792](#), [793](#)
- index\_t
  - fflas\_sparse.h, [629](#)
  - parallel.h, [777](#)
- InfNorm
  - FFLAS, [77](#)
- Info, [446](#), [447](#)
  - begin, [447](#), [448](#)
  - Info, [446–448](#)
  - operator=, [447](#), [448](#)
  - perm, [447](#), [448](#)
  - size, [447](#), [448](#)
- info
  - BlockTransposeSIMD< Field, Simd, >, [403](#)
- init
  - FieldSimd< \_Field >, [426](#), [427](#)
  - rns\_double, [465–467](#)
  - rns\_double\_extended, [477](#)
  - RNSInteger< RNS >, [481](#)
  - RNSIntegerMod< RNS >, [485](#), [486](#)
- init\_transpose
  - rns\_double, [466](#)
- init\_y
  - FFLAS::sparse\_details, [226](#)
- initA
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [453](#)
- initB

- MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [453](#)
- initC
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [453](#)
- initialize
  - ForStrategy1D< blocksize\_t, Cut, Param >, [433](#)
  - ForStrategy2D< blocksize\_t, Cut, Param >, [435](#)
- initOut
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [453](#)
- INLINE
  - fflas\_simd.h, [616](#)
- inplace
  - Bench< Elt >, [402](#)
- inputs
  - TestOneMethod< Simd >, [511](#)
- INST\_OR\_DECL
  - fflas\_L1\_inst.C, [707](#)
  - fflas\_L1\_inst.h, [708](#)
  - fflas\_L2\_inst.C, [711](#)
  - fflas\_L2\_inst.h, [712](#)
  - fflas\_L3\_inst.C, [714](#)
  - fflas\_L3\_inst.h, [715](#)
  - ffpack\_inst.C, [769](#)
  - ffpack\_inst.h, [770](#)
- integer
  - rns\_double, [464](#)
  - rns\_double\_extended, [476](#)
  - RNSInteger< RNS >, [480](#)
  - RNSIntegerMod< RNS >, [484](#)
- Interfaces, [42](#)
- interfaces.doxy, [695](#)
- inv
  - RNSIntegerMod< RNS >, [487](#)
- Invert
  - FFPACK, [320](#), [367](#)
- Invert2
  - FFPACK, [321](#), [367](#)
- Invert2\_modular\_double
  - ffpack.C, [742](#)
  - ffpack\_c.h, [762](#)
- Invert\_modular\_double
  - ffpack.C, [742](#)
  - ffpack\_c.h, [762](#)
- Invertin\_modular\_double
  - ffpack.C, [742](#)
  - ffpack\_c.h, [762](#)
- is\_all\_same< Args >, [448](#)
- is\_same\_element
  - Bench< Elt >, [400](#)
  - NoSimd< T >, [458](#)
  - Test< Elt >, [506](#)
- is\_simd< T >, [448](#)
  - type, [449](#)
  - value, [449](#)
- isMOne
  - RNSInteger< RNS >, [481](#)
- RNSIntegerMod< RNS >, [485](#)
- isOdd
  - FFPACK, [375](#)
- isOne
  - RNSInteger< RNS >, [480](#)
  - RNSIntegerMod< RNS >, [485](#)
- IsSingular
  - FFPACK, [325](#), [369](#)
- IsSingular\_modular\_double
  - ffpack.C, [743](#)
  - ffpack\_c.h, [763](#)
- isSparseMatrix< Field, M >, [449](#)
- isSparseMatrixMKLFormat< F, M >, [449](#)
- isSparseMatrixSimdFormat< F, M >, [449](#)
- isTerminated
  - ForStrategy1D< blocksize\_t, Cut, Param >, [433](#)
  - ForStrategy2D< blocksize\_t, Cut, Param >, [435](#)
- isZero
  - RNSInteger< RNS >, [481](#)
  - RNSIntegerMod< RNS >, [485](#)
- isZOSparseMatrix< F, M >, [450](#)
- Iterative, [450](#)
- iters
  - Bench< Elt >, [402](#)
- jbegin
  - ForStrategy2D< blocksize\_t, Cut, Param >, [435](#)
- jend
  - ForStrategy2D< blocksize\_t, Cut, Param >, [435](#)
- kaapi\_routines.inl, [776](#)
  - \_\_FFLASFFPACK\_KAAPI\_ROUTINES\_INL, [776](#)
- KellerGehrig
  - FFPACK::Protected, [391](#)
- KGFast
  - FFPACK::Protected, [392](#)
- KGFast\_generalized
  - FFPACK::Protected, [392](#)
- KrylovElim
  - FFPACK, [369](#)
- KrylovElim\_modular\_double
  - ffpack.C, [742](#)
  - ffpack\_c.h, [762](#)
- lapack.C, [797](#)
  - \_\_FFLASFFPACK\_CONFIGURATION, [797](#)
  - \_\_FFLASFFPACK\_HAVE\_LAPACK, [797](#)
  - main, [797](#)
- LAPACKPerm2MathPerm
  - FFPACK, [303](#)
  - ffpack.C, [732](#)
  - ffpack\_c.h, [753](#)
- lastBlockSize
  - ForStrategy1D< blocksize\_t, Cut, Param >, [434](#)
- lastCBS
  - ForStrategy2D< blocksize\_t, Cut, Param >, [437](#)
- lastRBS
  - ForStrategy2D< blocksize\_t, Cut, Param >, [437](#)
- launch\_fger

- test-fger.C, [813](#)
- launch\_fger\_dispatch
  - test-fger.C, [813](#)
- launch\_MM
  - test-fgemm.C, [809](#)
- launch\_MM\_dispatch
  - test-fgemm-check.C, [807](#)
  - test-fgemm.C, [809](#)
- launch\_MV
  - test-fgemv.C, [811](#)
- launch\_MV\_dispatch
  - test-fgemv.C, [811](#)
- launch\_test
  - test-charpoly.C, [799](#)
  - test-lu.C, [835](#)
  - test-quasisep.C, [842](#)
- launch\_wino
  - benchmark-wino.C, [547](#)
- LazyTag, [450](#)
- LD
  - fflas\_transpose.h, [658](#)
- LeadingSubmatrixRankProfiles
  - FFPACK, [331](#)
  - ffpack.C, [745](#)
  - ffpack\_c.h, [765](#)
- lesser
  - ScalFunctions< Element >, [494](#)
- lesser\_eq
  - ScalFunctions< Element >, [494](#)
- limits< T >, [450](#)
- LQUPtoInverseOfFullRankMinor
  - FFPACK, [345](#), [375](#)
- LT\_OBJDIR
  - config.h, [563](#)
- LTBruhatGen
  - FFPACK, [340](#)
- LTQSorter
  - FFPACK, [342](#)
- LUdivine
  - FFPACK, [314](#), [353](#), [354](#), [365](#)
- LUdivine\_construct
  - FFPACK::Protected, [390](#), [396](#)
- LUdivine\_gauss
  - FFPACK, [353](#), [366](#)
- LUdivine\_gauss\_modular\_double
  - ffpack\_c.h, [758](#)
- LUdivine\_modular\_double
  - ffpack.C, [736](#)
  - ffpack\_c.h, [757](#)
- LUdivine\_small
  - FFPACK, [353](#), [365](#)
- LUdivine\_small\_modular\_double
  - ffpack\_c.h, [758](#)
- LUKrylov
  - FFPACK::Protected, [392](#)
- LUKrylov\_KGFast
  - FFPACK::Protected, [393](#)
- m
  - Bench< Elt >, [402](#)
  - MachineFloatTag, [451](#)
  - MachineIntTag, [451](#)
  - main
    - 101-fgemm.C, [849](#)
    - 2x2-fgemm.C, [850](#)
    - 2x2-ftrsv.C, [850](#)
    - 2x2-pluq.C, [851](#)
    - arithprog.C, [517](#)
    - benchmark-charpoly-mp.C, [523](#)
    - benchmark-charpoly.C, [524](#)
    - benchmark-checkers.C, [525](#)
    - benchmark-dgemm.C, [526](#)
    - benchmark-dgetrf.C, [527](#)
    - benchmark-dgetri.C, [527](#)
    - benchmark-dsytrf.C, [528](#)
    - benchmark-dtrsm.C, [528](#)
    - benchmark-dtrtri.C, [529](#)
    - benchmark-fadd-lvl2.C, [529](#)
    - benchmark-fdot.C, [530](#)
    - benchmark-fgemm-mp.C, [531](#)
    - benchmark-fgemm-rns.C, [533](#)
    - benchmark-fgemm.C, [533](#)
    - benchmark-fgemv-mp.C, [534](#)
    - benchmark-fgemv.C, [538](#)
    - benchmark-fgesv.C, [538](#)
    - benchmark-fsyr2k.C, [539](#)
    - benchmark-fsyrk.C, [539](#)
    - benchmark-fsytrf.C, [540](#)
    - benchmark-ftrsm-mp.C, [541](#)
    - benchmark-ftrsm.C, [541](#)
    - benchmark-ftrsv.C, [542](#)
    - benchmark-ftrtri.C, [542](#)
    - benchmark-inverse.C, [543](#)
    - benchmark-lqup-mp.C, [543](#)
    - benchmark-lqup.C, [544](#)
    - benchmark-pluq.C, [545](#)
    - benchmark-quasisep.C, [546](#)
    - benchmark-storage-transpose.C, [547](#)
    - benchmark-wino.C, [548](#)
  - cblas.C, [795](#)
  - charpoly.C, [518](#)
  - clapack.C, [795](#)
  - cuda.C, [796](#)
  - det.C, [548](#)
  - fblas.C, [796](#)
  - fflas-101\_1.C, [851](#)
  - fflas-101\_3.C, [851](#)
  - fflas\_101.C, [852](#)
  - fflas\_101\_lvl1.C, [852](#)
  - ffpack-fgesv.C, [853](#)
  - ffpack-solve.C, [853](#)
  - fsyrk.C, [519](#)
  - fsytrf.C, [520](#)
  - ftrtri.C, [521](#)
  - lapack.C, [797](#)
  - matmul.C, [548](#)
  - pluq.C, [522](#)

- rank.C, [549](#)
- regression-check.C, [798](#)
- solve.C, [549](#)
- test-charpoly-check.C, [799](#)
- test-charpoly.C, [800](#)
- test-compressQ.C, [800](#)
- test-det-check.C, [801](#)
- test-det.C, [802](#)
- test-echelon.C, [804](#)
- test-fadd.C, [805](#)
- test-fdot.C, [806](#)
- test-fgemm-check.C, [808](#)
- test-fgemm.C, [810](#)
- test-fgemv.C, [812](#)
- test-fger.C, [813](#)
- test-fgesv.C, [815](#)
- test-finit.C, [816](#)
- test-fscal.C, [817](#)
- test-fsyr2k.C, [819](#)
- test-fsyrk.C, [821](#)
- test-fsytrf.C, [822](#)
- test-ftrmm.C, [823](#)
- test-ftrmv.C, [824](#)
- test-ftrsm-check.C, [825](#)
- test-ftrsm.C, [826](#)
- test-ftrssyr2k.C, [827](#)
- test-ftrstr.C, [828](#)
- test-ftrsv.C, [830](#)
- test-ftrtri.C, [831](#)
- test-interfaces-c.c, [831](#)
- test-invert-check.C, [832](#)
- test-io.C, [832](#)
- test-lu.C, [836](#)
- test-maxdelayeddim.C, [837](#)
- test-minpoly.C, [838](#)
- test-multifile2.C, [838](#)
- test-nullspace.C, [840](#)
- test-permutations.C, [840](#)
- test-pluq-check.C, [841](#)
- test-quasisep.C, [843](#)
- test-rankprofiles.C, [844](#)
- test-rpm.C, [844](#)
- test-simd.C, [848](#)
- test-solve.C, [848](#)
- test-storage-transpose.C, [849](#)
- winograd.C, [523](#)
- mainpage.doxy, [548](#)
- mask\_t
  - read\_sparse.h, [653](#)
- MatF2MatD\_Triangular
  - FFLAS::Protected, [222](#)
- MatF2MatFI\_Triangular
  - FFLAS::Protected, [222](#)
- MathPerm2LAPACKPerm
  - FFPACK, [303](#)
  - ffpack.C, [732](#)
  - ffpack\_c.h, [753](#)
- Matio.h, [793](#)
- read\_field, [793](#)
- write\_field, [793](#)
- matmul.C, [548](#)
  - main, [548](#)
- matmul.doxy, [583](#)
- Matrix Multiplication Algorithms, [42](#)
- MatrixApplyS
  - FFPACK, [355](#), [356](#)
- MatrixApplyS\_modular\_double
  - ffpack.C, [732](#)
  - ffpack\_c.h, [754](#)
- MatrixApplyT
  - FFPACK, [357](#)
- MatrixApplyT\_modular\_double
  - ffpack.C, [732](#)
  - ffpack\_c.h, [754](#)
- MatVecMinPoly
  - FFPACK, [324](#), [368](#)
  - FFPACK::Protected, [393](#)
- max3
  - FFLAS, [77](#)
- max4
  - FFLAS, [77](#)
- MAX\_THREADS
  - parallel.h, [778](#)
- MAX\_WITH\_SIZE\_T
  - test-maxdelayeddim.C, [837](#)
- maxCardinality
  - FFLAS, [154](#)
- maxCardinality< Givaro::Modular< int32\_t > >
  - FFLAS, [154](#)
- maxCardinality< Givaro::Modular< int64\_t > >
  - FFLAS, [154](#)
- maxCol
  - StatsMatrix, [502](#)
- maxColDifference
  - StatsMatrix, [503](#)
- MaxDelayedDim
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [453](#)
- maxElement
  - RNSIntegerMod< RNS >, [485](#)
- maxpy
  - FieldSimd< \_Field >, [430](#)
- maxpyin
  - FieldSimd< \_Field >, [430](#)
- maxRow
  - StatsMatrix, [502](#)
- maxRowDifference
  - StatsMatrix, [503](#)
- MaxStorableValue
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [456](#)
- MG\_DEFAULT
  - benchmark-fgemm-mp.C, [531](#)
  - benchmark-fgemv-mp.C, [534](#)
- min3
  - FFLAS, [77](#)

- min4
  - FFLAS, [77](#)
- min\_types
  - FFLAS::Protected, [219](#), [220](#)
- minCardinality
  - FFLAS, [154](#)
- minCol
  - StatsMatrix, [502](#)
- minColDifference
  - StatsMatrix, [503](#)
- minElement
  - RNSIntegerMod< RNS >, [485](#)
- MinPoly
  - FFPACK, [323](#), [368](#)
- minRow
  - StatsMatrix, [502](#)
- minRowDifference
  - StatsMatrix, [503](#)
- MKL\_CONFIG, [396](#)
- MKLSparseMatrixFormat
  - FFLAS, [73](#)
- MMHelper
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-  
Trait >, [452](#), [453](#)
- MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, [451](#)
  - Amax, [455](#)
  - Amin, [455](#)
  - Aunfit, [454](#)
  - Bmax, [455](#)
  - Bmin, [455](#)
  - Bunfit, [454](#)
  - checkA, [454](#)
  - checkB, [454](#)
  - checkOut, [454](#)
  - Cmax, [455](#)
  - Cmin, [455](#)
  - DelayedField, [452](#)
  - delayedField, [456](#)
  - DelayedField\_t, [452](#)
  - DFElt, [452](#)
  - FieldMax, [455](#)
  - FieldMin, [455](#)
  - initA, [453](#)
  - initB, [453](#)
  - initC, [453](#)
  - initOut, [453](#)
  - MaxDelayedDim, [453](#)
  - MaxStorableValue, [456](#)
  - MMHelper, [452](#), [453](#)
  - operator<=, [455](#)
  - Outmax, [456](#)
  - Outmin, [456](#)
  - parseq, [456](#)
  - recLevel, [455](#)
  - Self\_t, [452](#)
  - setOutBounds, [454](#)
- mod
  - FieldSimd< \_Field >, [428](#)
- MODE
  - parallel.h, [780](#)
- ModeTraits< Field >, [456](#)
  - value, [456](#)
- ModField
  - rns\_double, [464](#)
  - rns\_double\_extended, [476](#)
  - RNSIntegerMod< RNS >, [484](#)
- modp
  - FFLAS::vectorised, [282](#)
  - FFLAS::vectorised::unswitch, [284](#), [285](#)
- ModularBalanced< T >, [457](#)
- ModularTag, [457](#)
- mOne
  - RNSInteger< RNS >, [482](#)
  - RNSIntegerMod< RNS >, [489](#)
- mone
  - FFLAS, [76](#)
- MonotonicApplyP
  - FFPACK, [305](#)
- MonotonicCompress
  - FFPACK, [354](#)
- MonotonicCompressCycles
  - FFPACK, [354](#)
- MonotonicCompressMorePivots
  - FFPACK, [354](#)
- MonotonicExpand
  - FFPACK, [355](#)
- Montgomery< T >, [457](#)
- mul
  - FieldSimd< \_Field >, [428](#), [429](#)
  - RNSIntegerMod< RNS >, [487](#)
  - ScalFunctions< Element >, [493](#)
- mul\_r
  - FieldSimd< \_Field >, [429](#)
- mulin
  - FieldSimd< \_Field >, [429](#)
  - ScalFunctions< Element >, [493](#)
- mvcnt
  - test-lu.C, [836](#)
- n
  - Bench< Elt >, [402](#)
- name
  - TestOneMethod< Simd >, [511](#)
- nb\_lref
  - TestOneMethod< Simd >, [510](#)
- nDenseCols
  - StatsMatrix, [503](#)
- nDenseRows
  - StatsMatrix, [503](#)
- need\_field\_characteristic< Field >, [457](#)
  - value, [457](#)
- NeedDoublePreAddReduction
  - FFLAS::Protected, [214](#)
- NeedPreAddReduction
  - FFLAS::Protected, [213](#)
- NeedPreAxyReduction



- FFLAS::Protected, 219
- NeedPreScalReduction
  - FFLAS::Protected, 218, 219
- NeedPreSubReduction
  - FFLAS::Protected, 213
- neg
  - RNSIntegerMod< RNS >, 486
- nEmptyCols
  - StatsMatrix, 503
- nEmptyColsEnd
  - StatsMatrix, 503
- nEmptyRows
  - StatsMatrix, 503
- newD
  - FFPACK::Protected, 394
- NEWWINO
  - fgemm\_winograd.inl, 583
- nMOnes
  - StatsMatrix, 502
- nnz
  - StatsMatrix, 502
- none
  - HelperFlag, 445
- nOnes
  - StatsMatrix, 502
- nonsquare\_inplace\_v1
  - FFLAS::\_ftranspose\_impl, 190
- nonsquare\_inplace\_v2
  - FFLAS::\_ftranspose\_impl, 190
- NonZeroRandomMatrix
  - FFPACK, 375, 376
- NORML\_MOD
  - fflas\_simd.h, 616
- NoSimd< T >, 457
  - aligned\_allocator, 458
  - aligned\_vector, 458
  - alignment, 459
  - compliant, 458
  - is\_same\_element, 458
  - scalar\_t, 458
  - type\_string, 458
  - valid, 458
  - vect\_size, 459
  - vect\_t, 458
- NoSimdSparseMatrix
  - FFLAS, 73
- NOSPLIT
  - parallel.h, 782
- not\_inplace
  - FFLAS::\_ftranspose\_impl, 190
- nOthers
  - StatsMatrix, 502
- NotMKLSparseMatrixFormat
  - FFLAS, 73
- NotZOSparseMatrix
  - FFLAS, 72
- NullSpaceBasis
  - FFPACK, 328, 371
- NullSpaceBasis\_modular\_double
  - ffpack.C, 744
  - ffpack\_c.h, 764
- NUM\_THREADS
  - parallel.h, 778
- NUMARGS
  - parallel.h, 780
- number\_kind
  - FFLAS, 76
- numBlock
  - ForStrategy1D< blocksize\_t, Cut, Param >, 434
- numblocks
  - ForStrategy1D< blocksize\_t, Cut, Param >, 433
- numColBlock
  - ForStrategy2D< blocksize\_t, Cut, Param >, 438
- numRowBlock
  - ForStrategy2D< blocksize\_t, Cut, Param >, 437
- numthreads
  - Parallel< C, P >, 459
  - Sequential, 496
- one
  - FFLAS, 76
  - RNSInteger< RNS >, 482
  - RNSIntegerMod< RNS >, 489
- OPENBLAS\_NUM\_THREADS
  - config.h, 563
- operator!=
  - rns\_double\_elt\_cstptr, 472
  - rns\_double\_elt\_ptr, 474
- operator<
  - rns\_double\_elt\_cstptr, 472
  - rns\_double\_elt\_ptr, 474
- operator<<
  - Compose< H1, H2 >, 413
  - FFLAS, 151
  - ForStrategy2D< blocksize\_t, Cut, Param >, 436
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, 455
  - Parallel< C, P >, 460
  - Sequential, 496
  - test-fsytrf.C, 821
  - test-simd.C, 847
- operator()
  - callLUdivine\_small< Element >, 404
  - Failure, 423
  - readMyMachineType< Field, T >, 463
  - RNSInteger< RNS >::RandIter, 461
  - RNSIntegerMod< RNS >::RandIter, 462
  - rnsRandIter< RNS >, 490
  - tfn\_minus, 511
  - tfn\_minus\_eq, 512
  - tfn\_mul, 512
  - tfn\_mul\_eq, 512
  - tfn\_plus, 513
  - tfn\_plus\_eq, 513
- operator+
  - rns\_double\_elt\_cstptr, 471
  - rns\_double\_elt\_ptr, 474



- operator++
  - ForStrategy1D< blocksize\_t, Cut, Param >, [433](#)
  - ForStrategy2D< blocksize\_t, Cut, Param >, [436](#)
  - rns\_double\_elt\_cstptr, [471](#)
  - rns\_double\_elt\_ptr, [474](#)
- operator+=
  - rns\_double\_elt\_cstptr, [471](#)
  - rns\_double\_elt\_ptr, [474](#)
- operator-
  - rns\_double\_elt\_cstptr, [471](#)
  - rns\_double\_elt\_ptr, [474](#)
- operator--
  - rns\_double\_elt\_cstptr, [471](#)
  - rns\_double\_elt\_ptr, [474](#)
- operator-=
  - rns\_double\_elt\_cstptr, [471](#)
  - rns\_double\_elt\_ptr, [474](#)
- operator=
  - Coo< Field >, [416](#)
  - Coo< ValT, IdxT >, [415](#), [418](#)
  - FieldSimd< \_Field >, [426](#)
  - Info, [447](#), [448](#)
  - rns\_double\_elt\_cstptr, [471](#)
  - rns\_double\_elt\_ptr, [474](#)
- operator&
  - rns\_double\_elt, [469](#)
  - rns\_double\_elt\_cstptr, [471](#), [472](#)
  - rns\_double\_elt\_ptr, [473](#), [474](#)
- operator[]
  - rns\_double\_elt\_cstptr, [471](#)
  - rns\_double\_elt\_ptr, [473](#), [474](#)
- operator\*
  - rns\_double\_elt\_cstptr, [471](#)
  - rns\_double\_elt\_ptr, [473](#)
- other
  - FFLAS, [76](#)
  - rns\_double\_elt\_cstptr, [472](#)
  - rns\_double\_elt\_ptr, [475](#)
- Outmax
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [456](#)
- Outmin
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [456](#)
- outputs\_scalar
  - TestOneMethod< Simd >, [511](#)
- outputs\_simd
  - TestOneMethod< Simd >, [511](#)
- pack
  - ScalFunctions< Element >, [495](#)
- pack\_even
  - ScalFunctions< Element >, [495](#)
- pack\_lhs
  - FFLAS::details, [205](#)
- pack\_odd
  - ScalFunctions< Element >, [495](#)
- pack\_rhs
  - FFLAS::details, [205](#)
- PACKAGE
  - config.h, [563](#)
- PACKAGE\_BUGREPORT
  - config.h, [563](#)
- PACKAGE\_NAME
  - config.h, [563](#)
- PACKAGE\_STRING
  - config.h, [563](#)
- PACKAGE\_TARNAME
  - config.h, [563](#)
- PACKAGE\_URL
  - config.h, [563](#)
- PACKAGE\_VERSION
  - config.h, [563](#)
- PAR\_BLOCK
  - parallel.h, [777](#)
- Parallel
  - Parallel< C, P >, [459](#)
- Parallel< C, P >, [459](#)
  - Cut, [459](#)
  - numthreads, [459](#)
  - operator<<, [460](#)
  - Parallel, [459](#)
  - Param, [459](#)
  - set\_numthreads, [459](#)
- parallel.h, [776](#)
  - \_\_FFLASFFPACK\_SEQUENTIAL, [777](#)
  - BARRIER, [777](#)
  - BEGIN\_PARALLEL\_MAIN, [778](#)
  - CHECK\_DEPENDENCIES, [777](#)
  - COMMA, [780](#)
  - CONSTREFERENCE, [778](#)
  - END\_PARALLEL\_MAIN, [778](#)
  - FOR1D, [779](#)
  - FOR2D, [779](#)
  - FORBLOCK1D, [778](#)
  - FORBLOCK2D, [779](#)
  - index\_t, [777](#)
  - MAX\_THREADS, [778](#)
  - MODE, [780](#)
  - NOSPLIT, [782](#)
  - NUM\_THREADS, [778](#)
  - NUMARGS, [780](#)
  - PAR\_BLOCK, [777](#)
  - PARFOR1D, [779](#)
  - PARFOR2D, [780](#)
  - PARFORBLOCK1D, [779](#)
  - PARFORBLOCK2D, [780](#)
  - PP\_ARG\_N, [781](#)
  - PP\_NARG\_, [781](#)
  - PP\_RSEQ\_N, [782](#)
  - READ, [778](#)
  - READWRITE, [778](#)
  - RETURNPARAM, [780](#)
  - SET\_THREADS, [778](#)
  - splitt, [783](#)
  - SPLITTER, [783](#)
  - splitting\_0, [782](#)

- splitting\_1, [782](#)
- splitting\_2, [782](#)
- splitting\_3, [783](#)
- SYNCH\_GROUP, [777](#)
- TASK, [777](#)
- THREAD\_INDEX, [777](#)
- VALUE, [778](#)
- WAIT, [777](#)
- WRITE, [778](#)
- Param
  - Parallel< C, P >, [459](#)
- PARFOR1D
  - parallel.h, [779](#)
- PARFOR2D
  - parallel.h, [780](#)
- PARFORBLOCK1D
  - parallel.h, [779](#)
- PARFORBLOCK2D
  - parallel.h, [780](#)
- parseArguments
  - FFLAS, [185](#)
- parseq
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, [456](#)
  - TRSMHelper< ReclterTrait, ParSeqTrait >, [515](#)
- PBASECASE\_K
  - ffpack\_ppluq.inl, [687](#)
- pColumnEchelonForm
  - FFPACK, [316](#)
- pColumnEchelonForm\_modular\_double
  - ffpack.C, [739](#)
- pColumnEchelonForm\_modular\_float
  - ffpack.C, [740](#)
- pColumnEchelonForm\_modular\_int32\_t
  - ffpack.C, [741](#)
- pColumnRankProfile
  - FFPACK, [331](#)
- pDet
  - FFPACK, [326](#)
- perm
  - Info, [447](#), [448](#)
- PermApplyS
  - FFPACK, [356](#)
- PermApplyS\_double
  - ffpack.C, [732](#)
  - ffpack\_c.h, [754](#)
- PermApplyT
  - FFPACK, [358](#)
- PermApplyT\_double
  - ffpack.C, [733](#)
  - ffpack\_c.h, [754](#)
- 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, [783](#)
- pfgemv.inl, [784](#)
- pftrand
  - FFLAS, [181](#)
- pfreduce
  - FFLAS, [109](#)
- pfspmm
  - FFLAS::sparse\_details, [232–234](#)
  - FFLAS::sparse\_details\_impl, [249](#), [256](#), [257](#), [259–261](#), [271](#)
- pfspmm\_dispatch
  - FFLAS::sparse\_details, [231](#), [232](#)
- pfspmm\_mone
  - FFLAS::sparse\_details\_impl, [250](#)
- pfspmm\_one
  - FFLAS::sparse\_details\_impl, [250](#)
- pfspmm\_zo
  - FFLAS::sparse\_details\_impl, [261](#)
- pfspmv
  - FFLAS::sparse\_details, [234](#), [235](#)
  - FFLAS::sparse\_details\_impl, [251](#), [258](#), [261](#), [262](#), [267](#), [271–274](#)
- pfspmv\_mone
  - FFLAS::sparse\_details\_impl, [252](#), [262](#), [263](#), [268](#), [274](#)
- pfspmv\_one
  - FFLAS::sparse\_details\_impl, [251](#), [252](#), [262](#), [268](#), [274](#)
- pfspmv\_task
  - FFLAS::sparse\_details\_impl, [251](#)
- pfsub
  - FFLAS, [79](#)
- pfsubin
  - FFLAS, [80](#)
- pfzero
  - FFLAS, [181](#)
- PLUQ
  - FFPACK, [313](#), [314](#), [361](#), [362](#), [365](#)
- pluq.C, [521](#), [522](#)
- CUBE, [521](#)
- GFOPS, [521](#)
- main, [522](#)
- PLUQ\_basecaseCrout
  - FFPACK, [360](#)
- PLUQ\_basecaseV2
  - FFPACK, [360](#)
- PLUQ\_basecaseV3
  - FFPACK, [360](#)
- PLUQ\_modular\_double
  - ffpack.C, [736](#)
  - ffpack\_c.h, [757](#)

- PLUQtoEchelonPermutation
  - FFPACK, [340](#)
  - ffpack.C, [748](#)
  - ffpack\_c.h, [768](#)
- pm1
  - HelperFlag, [446](#)
- pMMH
  - TRSMHelper< ReclterTrait, ParSeqTrait >, [515](#)
- PP\_ARG\_N
  - parallel.h, [781](#)
- PP\_NARG\_
  - parallel.h, [781](#)
- PP\_RSEQ\_N
  - parallel.h, [782](#)
- pPLUQ
  - FFPACK, [313](#)
- pRank
  - FFPACK, [325](#)
- preamble
  - FFLAS, [186](#)
- precompute\_cst
  - rns\_double, [465](#)
  - rns\_double\_extended, [477](#)
- pReducedColumnEchelonForm
  - FFPACK, [318](#)
- pReducedColumnEchelonForm\_modular\_double
  - ffpack.C, [740](#)
- pReducedColumnEchelonForm\_modular\_float
  - ffpack.C, [740](#)
- pReducedColumnEchelonForm\_modular\_int32\_t
  - ffpack.C, [741](#)
- pReducedRowEchelonForm
  - FFPACK, [319](#)
- pReducedRowEchelonForm\_modular\_double
  - ffpack.C, [740](#)
- pReducedRowEchelonForm\_modular\_float
  - ffpack.C, [741](#)
- pReducedRowEchelonForm\_modular\_int32\_t
  - ffpack.C, [742](#)
- prefetch
  - FFLAS, [189](#)
- print
  - Failure, [423](#)
- printHelpMessage
  - args-parser.h, [786](#)
- printPolynomial
  - test-charpoly-check.C, [799](#)
- printvect
  - test-compressQ.C, [800](#)
- productBruhatxTS
  - FFPACK, [344](#), [347](#)
- pRowEchelonForm
  - FFPACK, [317](#)
- pRowEchelonForm\_modular\_double
  - ffpack.C, [739](#)
- pRowEchelonForm\_modular\_float
  - ffpack.C, [740](#)
- pRowEchelonForm\_modular\_int32\_t
  - ffpack.C, [741](#)
- pRowRankProfile
  - FFPACK, [330](#)
- PSeq
  - benchmark-fgemm-rns.C, [533](#)
- pSolve
  - FFPACK, [328](#)
- PTRSM\_HYBRID\_THRESHOLD
  - fflas\_pftrsm.inl, [615](#)
- PURE
  - fflas\_simd.h, [616](#)
- queryCacheSizes
  - FFLAS, [189](#)
- queryL1CacheSize
  - FFLAS, [189](#)
- queryTopLevelCacheSize
  - FFLAS, [189](#)
- RandInt
  - FFPACK, [379](#)
- RandIter
  - RNSInteger< RNS >::RandIter, [460](#)
  - RNSIntegerMod< RNS >::RandIter, [461](#)
- random
  - RNSInteger< RNS >::RandIter, [460](#), [461](#)
  - RNSIntegerMod< RNS >::RandIter, [462](#)
  - rnsRandIter< RNS >, [490](#)
- RandomIndexSubset
  - FFPACK, [380](#)
- RandomKrylovPrecond
  - FFPACK::Protected, [393](#)
- RandomLTQSMATRIXWithRankandQSOrder
  - FFPACK, [388](#)
- RandomLTQSRankProfileMatrix
  - FFPACK, [382](#)
- RandomMatrix
  - FFPACK, [376](#), [377](#)
  - test-fscal.C, [817](#)
- RandomMatrixWithDet
  - FFPACK, [387](#)
- RandomMatrixWithRank
  - FFPACK, [379](#), [380](#)
- RandomMatrixWithRankandRandomRPM
  - FFPACK, [385](#)
- RandomMatrixWithRankandRPM
  - FFPACK, [382](#), [383](#)
- RandomNullSpaceVector
  - FFPACK, [328](#), [345](#), [371](#)
- RandomNullSpaceVector\_modular\_double
  - ffpack.C, [744](#)
  - ffpack\_c.h, [764](#)
- RandomPermutation
  - FFPACK, [381](#)
- RandomRankProfileMatrix
  - FFPACK, [381](#)
- RandomSymmetricMatrix
  - FFPACK, [379](#)
- RandomSymmetricMatrixWithRankandRandomRPM

- FFPACK, [386](#)
- RandomSymmetricMatrixWithRankandRPM
  - FFPACK, [383](#), [384](#)
- RandomSymmetricRankProfileMatrix
  - FFPACK, [382](#)
- RandomTriangularMatrix
  - FFPACK, [377](#), [378](#)
- Rank
  - FFPACK, [324](#), [325](#), [369](#)
- rank.C, [549](#)
  - main, [549](#)
- Rank\_modular\_double
  - ffpack.C, [743](#)
  - ffpack\_c.h, [763](#)
- RankProfileFromLU
  - FFPACK, [331](#)
  - ffpack.C, [745](#)
  - ffpack\_c.h, [765](#)
- READ
  - parallel.h, [778](#)
- read\_field
  - Matio.h, [793](#)
- read\_sparse.h, [652](#)
  - DNS\_BIN\_VER, [653](#)
  - mask\_t, [653](#)
- readDnsFormat
  - FFLAS, [152](#)
- readMachineType
  - FFLAS, [152](#)
- ReadMatrix
  - FFLAS, [186](#)
- readMyMachineType< Field, T >, [462](#)
  - Element, [462](#)
  - Element\_ptr, [462](#)
  - operator(), [463](#)
- readOrRandomMatrixWithRankAndRandomRPM
  - test-nullspace.C, [839](#)
- readSmsFormat
  - FFLAS, [151](#)
- readSprFormat
  - FFLAS, [151](#)
- READWRITE
  - parallel.h, [778](#)
- Rec\_Initialize
  - benchmark-pluq.C, [545](#)
- RecInt, [396](#)
- recLevel
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-  
Trait >, [455](#)
- Recursive, [463](#)
- reduce
  - FFLAS::vectorised, [281](#), [282](#)
  - rns\_double, [466](#)
  - rns\_double\_extended, [477](#)
  - RNSInteger< RNS >, [481](#)
  - RNSIntegerMod< RNS >, [486](#)
- reduce\_modp
  - RNSIntegerMod< RNS >, [487](#), [488](#)
- reduce\_modp\_rnsmajor
  - RNSIntegerMod< RNS >, [488](#)
- ReducedColumnEchelonForm
  - FFPACK, [317](#), [318](#), [367](#)
- ReducedColumnEchelonForm\_modular\_double
  - ffpack.C, [737](#)
  - ffpack\_c.h, [760](#)
- ReducedColumnEchelonForm\_modular\_float
  - ffpack.C, [738](#)
  - ffpack\_c.h, [760](#)
- ReducedColumnEchelonForm\_modular\_int32\_t
  - ffpack.C, [739](#)
  - ffpack\_c.h, [761](#)
- ReducedRowEchelonForm
  - FFPACK, [319](#), [366](#)
- ReducedRowEchelonForm2\_modular\_double
  - ffpack\_c.h, [761](#)
- ReducedRowEchelonForm\_modular\_double
  - ffpack.C, [737](#)
  - ffpack\_c.h, [760](#)
- ReducedRowEchelonForm\_modular\_float
  - ffpack.C, [738](#)
  - ffpack\_c.h, [760](#)
- ReducedRowEchelonForm\_modular\_int32\_t
  - ffpack.C, [739](#)
  - ffpack\_c.h, [761](#)
- REF\_modular\_double
  - ffpack\_c.h, [761](#)
- regression-check.C, [797](#)
  - check1, [797](#)
  - check2, [797](#)
  - check3, [798](#)
  - check4, [798](#)
  - checkZeroDimCharpoly, [798](#)
  - checkZeroDimMinPoly, [798](#)
  - gf2ModularBalanced, [798](#)
  - main, [798](#)
- Residu
  - Bench< Elt >, [400](#)
  - Test< Elt >, [506](#)
- RETURNPARAM
  - parallel.h, [780](#)
- ring
  - RNSInteger< RNS >::RandIter, [461](#)
  - RNSIntegerMod< RNS >::RandIter, [462](#)
  - rnsRandIter< RNS >, [490](#)
- rint< K >, [463](#)
- RNS, [43](#)
  - benchmark-fgemm-rns.C, [532](#)
- rns
  - RNSInteger< RNS >, [480](#)
  - RNSIntegerMod< RNS >, [484](#)
- rns-double-elt.h, [691](#)
- rns-double-recint.inl, [691](#)
  - \_\_FFLASFFPACK\_field\_rns\_double\_recint\_INL,  
[692](#)
- rns-double.h, [692](#)
- ROUND\_DOWN, [692](#)

- rns-double.inl, 693
  - \_\_FFLASFFPACK\_field\_rns\_double\_INL, 693
- rns-integer-mod.h, 693
- rns-integer.h, 694
- rns.h, 694
- rns.inl, 695
  - \_\_FFLASFFPACK\_field\_rns\_INL, 695
- rns\_double, 463
  - \_M, 467
  - \_MMi, 468
  - \_Mi, 467
  - \_basis, 467
  - \_basisMax, 467
  - \_crt\_in, 468
  - \_crt\_out, 468
  - \_field\_rns, 467
  - \_invbasis, 467
  - \_ldm, 468
  - \_mi\_sum, 468
  - \_negbasis, 467
  - \_pbits, 468
  - \_size, 468
  - BasisElement, 465
  - ConstElement\_ptr, 465
  - convert, 466, 467
  - convert\_transpose, 466
  - Element, 465
  - Element\_ptr, 465
  - init, 465–467
  - init\_transpose, 466
  - integer, 464
  - ModField, 464
  - precompute\_cst, 465
  - reduce, 466
  - rns\_double, 465
- rns\_double\_elt, 468
  - \_alloc, 469
  - \_ptr, 469
  - \_stride, 469
  - ~rns\_double\_elt, 469
  - operator&, 469
  - rns\_double\_elt, 469
- rns\_double\_elt\_cstptr, 470
  - \_alloc, 472
  - \_ptr, 472
  - \_stride, 472
  - operator!=, 472
  - operator<, 472
  - operator+, 471
  - operator++, 471
  - operator+==, 471
  - operator-, 471
  - operator--, 471
  - operator-=, 471
  - operator=, 471
  - operator&, 471, 472
  - operator[], 471
  - operator\*, 471
  - other, 472
  - rns\_double\_elt\_cstptr, 470, 471
- rns\_double\_elt\_ptr, 472
  - \_alloc, 475
  - \_ptr, 475
  - \_stride, 475
  - operator!=, 474
  - operator<, 474
  - operator+, 474
  - operator++, 474
  - operator+==, 474
  - operator-, 474
  - operator--, 474
  - operator-=, 474
  - operator=, 474
  - operator&, 473, 474
  - operator[], 473, 474
  - operator\*, 473
  - other, 475
  - rns\_double\_elt\_ptr, 473
- rns\_double\_extended, 475
  - \_M, 478
  - \_MMi, 478
  - \_Mi, 478
  - \_basis, 478
  - \_basisMax, 478
  - \_crt\_in, 478
  - \_crt\_out, 478
  - \_field\_rns, 478
  - \_invbasis, 478
  - \_ldm, 478
  - \_negbasis, 478
  - \_pbits, 478
  - \_size, 478
  - BasisElement, 476
  - ConstElement\_ptr, 476
  - convert, 477
  - Element, 476
  - Element\_ptr, 476
  - init, 477
  - integer, 476
  - ModField, 476
  - precompute\_cst, 477
  - reduce, 477
  - rns\_double\_extended, 476
- RNSElementTag, 479
- RNSInteger
  - RNSInteger< RNS >, 480
- RNSInteger< RNS >, 479
  - \_rns, 482
  - assign, 482
  - BasisElement, 480
  - cardinality, 481
  - characteristic, 481
  - ConstElement\_ptr, 480
  - convert, 481
  - Element, 480
  - Element\_ptr, 480

- init, [481](#)
- integer, [480](#)
- isMOne, [481](#)
- isOne, [480](#)
- isZero, [481](#)
- mOne, [482](#)
- one, [482](#)
- reduce, [481](#)
- rns, [480](#)
- RNSInteger, [480](#)
- size, [480](#)
- write, [482](#)
- zero, [482](#)
- RNSInteger< RNS >::RandIter, [460](#)
  - operator(), [461](#)
  - RandIter, [460](#)
  - random, [460](#), [461](#)
  - ring, [461](#)
- RNSIntegerMod
  - RNSIntegerMod< RNS >, [484](#)
- RNSIntegerMod< RNS >, [482](#)
  - \_F, [488](#)
  - \_Mi\_modp\_rns, [488](#)
  - \_RNSdelayed, [489](#)
  - \_iM\_modp\_rns, [488](#)
  - \_p, [488](#)
  - \_rns, [488](#)
  - add, [486](#)
  - areEqual, [487](#)
  - assign, [486](#)
  - axpyin, [487](#)
  - BasisElement, [484](#)
  - cardinality, [485](#)
  - characteristic, [485](#)
  - ConstElement\_ptr, [484](#)
  - convert, [486](#)
  - delayed, [484](#)
  - Element, [484](#)
  - Element\_ptr, [484](#)
  - init, [485](#), [486](#)
  - integer, [484](#)
  - inv, [487](#)
  - isMOne, [485](#)
  - isOne, [485](#)
  - isZero, [485](#)
  - maxElement, [485](#)
  - minElement, [485](#)
  - ModField, [484](#)
  - mOne, [489](#)
  - mul, [487](#)
  - neg, [486](#)
  - one, [489](#)
  - reduce, [486](#)
  - reduce\_modp, [487](#), [488](#)
  - reduce\_modp\_rnsmajor, [488](#)
  - rns, [484](#)
  - RNSIntegerMod, [484](#)
  - size, [484](#)
  - sub, [486](#)
  - write, [487](#)
  - write\_matrix, [487](#)
  - write\_matrix\_long, [488](#)
  - zero, [489](#)
- RNSIntegerMod< RNS >::RandIter, [461](#)
  - operator(), [462](#)
  - RandIter, [461](#)
  - random, [462](#)
  - ring, [462](#)
- RNSModulus
  - FFLAS::CuttingStrategy, [198](#)
- rnsRandIter
  - rnsRandIter< RNS >, [489](#)
- rnsRandIter< RNS >, [489](#)
  - operator(), [490](#)
  - random, [490](#)
  - ring, [490](#)
  - rnsRandIter, [489](#)
- ROUND\_DOWN
  - fflas\_sparse.h, [629](#)
  - rns-double.h, [692](#)
- Row, [490](#)
- row
  - Coo< Field >, [417](#)
  - Coo< ValT, IdxT >, [415](#), [418](#)
- rowblockindex
  - ForStrategy2D< blocksize\_t, Cut, Param >, [436](#)
- rowBlockSize
  - ForStrategy2D< blocksize\_t, Cut, Param >, [437](#)
- rowdim
  - StatsMatrix, [502](#)
- RowEchelonForm
  - FFPACK, [316](#), [317](#), [366](#)
- RowEchelonForm\_modular\_double
  - ffpack.C, [736](#)
  - ffpack\_c.h, [758](#)
- RowEchelonForm\_modular\_float
  - ffpack.C, [737](#)
  - ffpack\_c.h, [759](#)
- RowEchelonForm\_modular\_int32\_t
  - ffpack.C, [738](#)
  - ffpack\_c.h, [759](#)
- rownumblocks
  - ForStrategy2D< blocksize\_t, Cut, Param >, [436](#)
- RowRankProfile
  - FFPACK, [329](#), [330](#), [371](#)
- RowRankProfile\_modular\_double
  - ffpack.C, [745](#)
  - ffpack\_c.h, [764](#)
- RowRankProfileSubmatrix
  - FFPACK, [333](#), [372](#)
- RowRankProfileSubmatrix\_modular\_double
  - ffpack.C, [746](#)
  - ffpack\_c.h, [765](#)
- RowRankProfileSubmatrixIndices
  - FFPACK, [332](#), [372](#)
- RowRankProfileSubmatrixIndices\_modular\_double

- ffpack.C, 745
- ffpack\_c.h, 765
- ruint< K >, 490
- run
  - Bench< Elt >, 401
  - Test< Elt >, 507
- run\_with\_field
  - benchmark-charpoly.C, 524
  - benchmark-fdot.C, 530
  - benchmark-quasisep.C, 546
  - test-charpoly.C, 799
  - test-echelon.C, 804
  - test-fdot.C, 806
  - test-fgemm-check.C, 807
  - test-fgemm.C, 810
  - test-fgemv.C, 811
  - test-fger.C, 813
  - test-fgesv.C, 814
  - test-finit.C, 815
  - test-fsyr2k.C, 818
  - test-fsyrk.C, 820
  - test-fsytrf.C, 822
  - test-ftrmm.C, 823
  - test-ftrmv.C, 824
  - test-ftrsm.C, 826
  - test-ftrssyr2k.C, 827
  - test-ftrstr.C, 828
  - test-ftrsv.C, 829
  - test-ftrtri.C, 831
  - test-io.C, 832
  - test-lu.C, 836
  - test-minpoly.C, 838
  - test-nullspace.C, 840
  - test-quasisep.C, 843
  - test-rankprofiles.C, 843
  - test-solve.C, 848
- run\_with\_Integer
  - test-fdot.C, 806
- saxpy\_
  - config-blas.h, 556
- ScalAndReduce
  - FFLAS::Protected, 214, 218
- scalar\_t
  - FieldSimd< \_Field >, 426
  - NoSimd< T >, 458
- ScalFunctions< Element >, 490
  - add, 492
  - addin, 492
  - blend, 495
  - div, 493
  - eq, 494
  - fmadd, 493
  - fmaddin, 493
  - fmsub, 493
  - fmsubin, 493
  - fnmadd, 493
  - fnmaddin, 494
  - genInputs, 491
  - genInputsWithZero, 491
  - greater, 494
  - greater\_eq, 494
  - lesser, 494
  - lesser\_eq, 494
  - mul, 493
  - mulin, 493
  - pack, 495
  - pack\_even, 495
  - pack\_odd, 495
  - sub, 492
  - subin, 492
  - unpackhi, 494
  - unpacklo, 494
  - unpacklohi, 495
  - vand, 492
  - vandnot, 492
  - vectElt, 491
  - vor, 492
  - vxor, 492
  - zero, 492
- ScalFunctionsBase< Element, Enable >, 495
- scalp
  - FFLAS::vectorised, 282, 283
  - FFLAS::vectorised::unswitch, 285
- schedule\_bini.inl, 583
  - \_\_FFLASFFPACK\_fgemm\_bini\_INL, 583
- schedule\_winograd.inl, 583
  - \_\_FFLASFFPACK\_fgemm\_winograd\_INL, 584
- schedule\_winograd\_acc.inl, 584
  - \_\_FFLASFFPACK\_fgemm\_winograd\_acc\_INL, 585
- schedule\_winograd\_acc\_ip.inl, 585
  - \_\_FFLASFFPACK\_fgemm\_winograd\_acc\_ip\_INL, 585
- schedule\_winograd\_ip.inl, 585
  - \_\_FFLASFFPACK\_fgemm\_winograd\_ip\_INL, 586
- scopy\_
  - config-blas.h, 558
- sdot\_
  - config-blas.h, 557
- second\_component
  - Compose< H1, H2 >, 413
- Self
  - Coo< ValT, IdxT >, 414, 417
- Self\_t
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, 452
- SELL
  - FFLAS, 76
- sell.h, 653
- sell\_pspmv.inl, 653
  - \_\_FFLASFFPACK\_fflas\_sparse\_sell\_pspmv\_INL, 654
- sell\_spmv.inl, 654
  - \_\_FFLASFFPACK\_fflas\_sparse\_sell\_spmv\_INL, 655
- sell\_utils.inl, 655

- \_\_FFLASFFPACK\_fflas\_sparse\_sell\_utils\_INL, 655
- SELL\_ZO
  - FFLAS, 76
- Sequential, 496
  - numthreads, 496
  - operator<<, 496
  - Sequential, 496
- set\_numthreads
  - Parallel< C, P >, 459
- SET\_THREADS
  - parallel.h, 778
- setErrorStream
  - Failure, 423
- setOutBounds
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, 454
- sgemm\_
  - config-blas.h, 560
- sgemv\_
  - config-blas.h, 557
- sger\_
  - config-blas.h, 558
- Simd
  - fflas\_simd.h, 617
- simd
  - FieldSimd< \_Field >, 426
- SIMD wrapper, 42
- simd.doxy, 617
- Simd128
  - simd128.inl, 618
- simd128.inl, 617
  - \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_INL, 617
  - Simd128, 618
- simd128\_double.inl, 618
  - \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_double\_INL, 618
  - \_\_FFLASFFPACK\_simd512\_double\_INL, 624
- simd128\_float.inl, 618
  - \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_float\_INL, 618
- Simd128\_impl< ArithType, Int, Signed, Size >, 497
- simd128\_int16.inl, 618
  - \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_int16\_INL, 619
  - \_simd128\_int64\_INL, 625
- simd128\_int32.inl, 619
  - \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_int32\_INL, 619
- simd128\_int64.inl, 619
  - \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_int64\_INL, 620
- vect\_t, 620
- Simd128i\_base, 497
  - sll128, 497
  - srl128, 497
  - vand, 497
  - vandnot, 498
  - vect\_t, 497
- vor, 498
- vxor, 498
- zero, 497
- Simd256
  - simd256.inl, 620
- simd256.inl, 620
  - \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_INL, 620
  - Simd256, 620
- simd256\_double.inl, 621
  - \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_double\_INL, 621
- simd256\_float.inl, 621
  - \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_float\_INL, 621
- Simd256\_impl< ArithType, Int, Signed, Size >, 498
- simd256\_int16.inl, 621
  - \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_int16\_INL, 622
- simd256\_int32.inl, 622
  - \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_int32\_INL, 622
- simd256\_int64.inl, 622
  - \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_int64\_INL, 623
- vect\_t, 623
- Simd256fp\_base, 498
- Simd256i\_base, 498
  - vect\_t, 499
  - zero, 499
- Simd512
  - simd512.inl, 623
- simd512.inl, 623
  - \_\_FFLASFFPACK\_simd512\_INL, 623
  - Simd512, 623
- simd512\_double.inl, 623
  - \_\_FFLASFFPACK\_simd512\_double\_INL, 624
- simd512\_float.inl, 624
  - \_\_FFLASFFPACK\_simd512\_float\_INL, 624
- Simd512\_impl< ArithType, Int, Signed, Size >, 499
- simd512\_int32.inl, 624
  - \_\_FFLASFFPACK\_simd512\_int32\_INL, 624
- simd512\_int64.inl, 625
  - vect\_t, 625
- Simd512i\_base, 499
  - vand, 500
  - vandnot, 500
  - vect\_t, 499
  - vor, 500
  - vxor, 500
  - zero, 500
- SIMD\_INT
  - fflas\_simd.h, 616
- simd\_modular.inl, 625
- SimdChooser< T, bool, bool >, 500
- SimdSparseMatrix
  - FFLAS, 72



simdToType< T >, 500  
 Single, 500  
 size  
     BlockTransposeSIMD< Field, Simd, >, 403  
     Info, 447, 448  
     RNSInteger< RNS >, 480  
     RNSIntegerMod< RNS >, 484  
 SIZEOF\_\_\_INT64\_T  
     config.h, 564  
 SIZEOF\_CHAR  
     config.h, 563  
 SIZEOF\_INT  
     config.h, 563  
 SIZEOF\_LONG  
     config.h, 563  
 SIZEOF\_LONG\_LONG  
     config.h, 564  
 SIZEOF\_SHORT  
     config.h, 564  
 sll128  
     Simd128i\_base, 497  
 Solve  
     FFPACK, 327, 370  
 solve.C, 549  
     main, 549  
 Solve\_modular\_double  
     ffpack.C, 743  
     ffpack\_c.h, 763  
 solveLB  
     FFPACK, 346, 370  
 solveLB2  
     FFPACK, 346, 370  
 solveLB2\_modular\_double  
     ffpack.C, 744  
     ffpack\_c.h, 764  
 solveLB\_modular\_double  
     ffpack.C, 744  
     ffpack\_c.h, 763  
 Sparse< Field, SparseMatrix\_t, IdxT, PtrT >, 500  
 sparse\_delete  
     FFLAS, 146–150, 153  
 sparse\_init  
     FFLAS, 145–150, 153  
 sparse\_matrix\_traits.h, 656  
 sparse\_print  
     FFLAS, 147, 150, 153  
 SparseMatrix\_t  
     FFLAS, 76  
 SpecRankProfile  
     FFPACK, 369  
 SpecRankProfile\_modular\_double  
     ffpack.C, 743  
     ffpack\_c.h, 762  
 splitt  
     parallel.h, 783  
 SPLITTER  
     parallel.h, 783  
 splitting\_0  
     parallel.h, 782  
 splitting\_1  
     parallel.h, 782  
 splitting\_2  
     parallel.h, 782  
 splitting\_3  
     parallel.h, 783  
 SpMat< Field, flag >, 501  
     \_coo, 501  
     \_csr, 501  
     \_ell, 501  
 square\_inplace  
     FFLAS::\_ftranspose\_impl, 190  
 srl128  
     Simd128i\_base, 497  
 sscal\_  
     config-blas.h, 559  
 ST  
     fflas\_transpose.h, 658  
 StatsMatrix, 501  
     averageCol, 502  
     averageColDifference, 503  
     averageRow, 502  
     averageRowDifference, 503  
     coldim, 502  
     denseCols, 504  
     denseRows, 503  
     deviationCol, 503  
     deviationColDifference, 503  
     deviationRow, 502  
     deviationRowDifference, 503  
     maxCol, 502  
     maxColDifference, 503  
     maxRow, 502  
     maxRowDifference, 503  
     minCol, 502  
     minColDifference, 503  
     minRow, 502  
     minRowDifference, 503  
     nDenseCols, 503  
     nDenseRows, 503  
     nEmptyCols, 503  
     nEmptyColsEnd, 503  
     nEmptyRows, 503  
     nMOnes, 502  
     nnz, 502  
     nOnes, 502  
     nOthers, 502  
     rowdim, 502  
 STD\_RECINT\_SIZE  
     benchmark-fgemm-mp.C, 531  
     benchmark-fgemv-mp.C, 534  
 STDC\_HEADERS  
     config.h, 564  
 string, 504  
 string::const\_iterator, 413  
 string::const\_reverse\_iterator, 413  
 string::iterator, 450

- string::reverse\_iterator, 463
- strmm\_
  - config-blas.h, 559
- strsm\_
  - config-blas.h, 559
- sub
  - FFLAS::vectorised, 280
  - FieldSimd< \_Field >, 427, 428
  - RNSIntegerMod< RNS >, 486
  - ScalFunctions< Element >, 492
- sub\_r
  - FieldSimd< \_Field >, 428
- subin
  - FieldSimd< \_Field >, 428
  - ScalFunctions< Element >, 492
- subin\_r
  - FieldSimd< \_Field >, 428
- subp
  - FFLAS::vectorised, 280
- support\_fast\_mod< T >, 504
- support\_simd< T >, 504
- support\_simd\_add< T >, 505
- support\_simd\_mod< T >, 505
- swapval
  - FFPACK, 381
- SYNCH\_GROUP
  - parallel.h, 777
- SysTimer
  - FFLAS, 74
- TASK
  - parallel.h, 777
- tBC
  - test-lu.C, 836
  - test-permutations.C, 841
- Test
  - Test< Elt >, 507
- test
  - test-maxdelayeddim.C, 837
- Test< Elt >, 505
  - \_mm, 508
  - \_nn, 508
  - cardinality, 507
  - doTests, 507
  - Elt\_ptr, 506
  - enable\_if\_no\_simd\_t, 506
  - enable\_if\_simd128\_t, 506
  - enable\_if\_simd256\_t, 506
  - enable\_if\_simd512\_t, 507
  - enable\_if\_t, 506
  - F, 508
  - Field, 506
  - is\_same\_element, 506
  - Residu, 506
  - run, 507
  - Test, 507
  - test\_ftranspose, 507
- test-charpoly-check.C, 798
  - ENABLE\_CHECKER\_charpoly, 798
- main, 799
- printPolynomial, 799
- TIME\_CHECKER\_CHARPOLY, 798
- test-charpoly.C, 799
  - launch\_test, 799
  - main, 800
  - run\_with\_field, 799
- test-compressQ.C, 800
  - Field, 800
  - main, 800
  - printvect, 800
- test-det-check.C, 801
  - ENABLE\_CHECKER\_Det, 801
  - main, 801
  - TIME\_CHECKER\_Det, 801
- test-det.C, 801
  - main, 802
  - test\_det, 802
- test-echelon.C, 802
  - \_\_FFLASFFPACK\_GAUSSJORDAN\_BASECASE, 803
  - \_\_FFLASFFPACK\_PLUQ\_THRESHOLD, 803
  - \_\_FFLASFFPACK\_SEQUENTIAL, 803
  - main, 804
  - run\_with\_field, 804
  - test\_colechelon, 803
  - test\_redcoechelon, 803
  - test\_redrowechelon, 803
  - test\_rowechelon, 803
- test-fadd.C, 804
  - main, 805
  - test\_fadd, 805
  - test\_faddin, 805
  - test\_fsub, 805
  - test\_fsubin, 805
- test-fdot.C, 805
  - check\_fdot, 806
  - ENABLE\_ALL\_CHECKINGS, 806
  - main, 806
  - run\_with\_field, 806
  - run\_with\_Integer, 806
- test-fgemm-check.C, 807
  - ENABLE\_ALL\_CHECKINGS, 807
  - launch\_MM\_dispatch, 807
  - main, 808
  - run\_with\_field, 807
- test-fgemm.C, 808
  - check\_MM, 809
  - ENABLE\_CHECKER\_fgemm, 809
  - launch\_MM, 809
  - launch\_MM\_dispatch, 809
  - main, 810
  - run\_with\_field, 810
- test-fgemv.C, 810
  - check\_MV, 811
  - launch\_MV, 811
  - launch\_MV\_dispatch, 811
  - main, 812

- run\_with\_field, 811
- test-fger.C, 812
  - check\_fger, 813
  - launch\_fger, 813
  - launch\_fger\_dispatch, 813
  - main, 813
  - run\_with\_field, 813
  - TIME, 812
- test-fgesv.C, 814
  - main, 815
  - run\_with\_field, 814
  - test\_rect\_fgesv, 814
  - test\_square\_fgesv, 814
- test-finit.C, 815
  - main, 816
  - run\_with\_field, 815
  - test\_freduce, 815
- test-fscal.C, 816
  - main, 817
  - RandomMatrix, 817
  - test\_fscal, 816, 817
  - test\_fscaln, 817
- test-fsyr2k.C, 818
  - check\_fsyr2k, 818
  - ENABLE\_ALL\_CHECKINGS, 818
  - main, 819
  - run\_with\_field, 818
- test-fsyrc.C, 819
  - check\_computeS1S2, 820
  - check\_fsyrc, 820
  - check\_fsyrc\_bkdiag, 820
  - check\_fsyrc\_diag, 820
  - ENABLE\_ALL\_CHECKINGS, 820
  - main, 821
  - run\_with\_field, 820
- test-fsytrf.C, 821
  - main, 822
  - operator<<, 821
  - run\_with\_field, 822
  - test\_generic\_fsytrf, 822
  - test\_RPM\_fsytrf, 821
- test-ftmm.C, 822
  - \_\_FFLASFFPACK\_SEQUENTIAL, 823
  - check\_ftmm, 823
  - main, 823
  - run\_with\_field, 823
- test-ftmv.C, 823
  - \_\_FFLASFFPACK\_SEQUENTIAL, 824
  - check\_ftmv, 824
  - ENABLE\_ALL\_CHECKINGS, 824
  - main, 824
  - run\_with\_field, 824
- test-ftsm-check.C, 825
  - ENABLE\_ALL\_CHECKINGS, 825
  - main, 825
- test-ftsm.C, 825
  - \_\_FFLASFFPACK\_SEQUENTIAL, 826
  - check\_ftsm, 826
- ENABLE\_ALL\_CHECKINGS, 826
  - main, 826
  - run\_with\_field, 826
- test-ftssyr2k.C, 826
  - check\_ftssyr2k, 827
  - ENABLE\_ALL\_CHECKINGS, 827
  - main, 827
  - run\_with\_field, 827
- test-ftstr.C, 828
  - check\_ftstr, 828
  - ENABLE\_ALL\_CHECKINGS, 828
  - main, 828
  - run\_with\_field, 828
- test-ftsv.C, 829
  - \_\_FFLASFFPACK\_SEQUENTIAL, 829
  - check\_ftsv, 829
  - ENABLE\_ALL\_CHECKINGS, 829
  - main, 830
  - run\_with\_field, 829
- test-fttri.C, 830
  - \_\_FFLASFFPACK\_SEQUENTIAL, 830
  - check\_fttri, 830
  - ENABLE\_ALL\_CHECKINGS, 830
  - main, 831
  - run\_with\_field, 831
- test-interfaces-c.c, 831
  - main, 831
- test-invert-check.C, 831
  - ENABLE\_ALL\_CHECKINGS, 832
  - main, 832
- test-io.C, 832
  - main, 832
  - run\_with\_field, 832
- test-lu.C, 833
  - \_\_FFLASFFPACK\_SEQUENTIAL, 833
  - \_\_LUDIVINE\_CUTOFF, 834
  - BASECASE\_K, 833
  - launch\_test, 835
  - main, 836
  - mvcnt, 836
  - run\_with\_field, 836
  - tBC, 836
  - test\_LUdivine, 834
  - test\_pluq, 835
  - tgemm, 836
  - timtot, 836
  - tperm, 836
  - trest, 836
  - ttrsm, 836
  - verifPLUQ, 834
- test-maxdelayeddim.C, 837
  - main, 837
  - MAX\_WITH\_SIZE\_T, 837
  - test, 837
- test-minpoly.C, 837
  - check\_minpoly, 838
  - main, 838
  - run\_with\_field, 838

- test-multifile1.C, 838
- test-multifile2.C, 838
  - main, 838
- test-nullspace.C, 839
  - checkingMessage, 839
  - main, 840
  - readOrRandomMatrixWithRankAndRandomRPM, 839
  - run\_with\_field, 840
  - test\_nullspace, 839
- test-permutations.C, 840
  - checkMonotonicApplyP, 840
  - main, 840
  - tBC, 841
  - tgemm, 841
  - timtot, 841
  - tperm, 841
  - trest, 841
  - ttrsm, 841
- test-pluq-check.C, 841
  - ENABLE\_ALL\_CHECKINGS, 841
  - main, 841
- test-quasisep.C, 842
  - launch\_test, 842
  - main, 843
  - run\_with\_field, 843
  - test\_BruhatGenerator, 842
  - testLTQSRPM, 842
- test-rankprofiles.C, 843
  - \_\_FFLASFFPACK\_SEQUENTIAL, 843
  - main, 844
  - run\_with\_field, 843
- test-rpm.C, 844
  - checkRPM, 844
  - checkSymmetricRPM, 844
  - main, 844
- test-simd.C, 844
  - \_TEST\_ONE, 846
  - check\_eq, 847
  - cmp, 847
  - eval\_func\_on\_array, 847
  - main, 848
  - operator<<, 847
  - TEST\_IMPL, 846
  - test\_impl, 848
  - test\_impl\_base, 847
  - TEST\_ONE\_OP, 846
  - TEST\_ONE\_OP\_WZ, 846
- test-solve.C, 848
  - check\_solve, 848
  - main, 848
  - run\_with\_field, 848
- test-storage-transpose.C, 849
  - main, 849
- test-utils.h, 793
- test\_BruhatGenerator
  - test-quasisep.C, 842
- test\_colechelon
  - test-echelon.C, 803
- test\_det
  - test-det.C, 802
- test\_fadd
  - test-fadd.C, 805
- test\_faddin
  - test-fadd.C, 805
- test\_freduce
  - test-finit.C, 815
- test\_fscal
  - test-fscal.C, 816, 817
- test\_fscalin
  - test-fscal.C, 817
- test\_fsub
  - test-fadd.C, 805
- test\_fsubin
  - test-fadd.C, 805
- test\_ftranspose
  - Test< Elt >, 507
- test\_generic\_fsytrf
  - test-fsytrf.C, 822
- TEST\_IMPL
  - test-simd.C, 846
- test\_impl
  - test-simd.C, 848
- test\_impl\_base
  - test-simd.C, 847
- test\_LUdivine
  - test-lu.C, 834
- test\_nullspace
  - test-nullspace.C, 839
- TEST\_ONE\_OP
  - test-simd.C, 846
- TEST\_ONE\_OP\_WZ
  - test-simd.C, 846
- test\_pluq
  - test-lu.C, 835
- test\_rect\_fgesv
  - test-fgesv.C, 814
- test\_redcoechelon
  - test-echelon.C, 803
- test\_redrowechelon
  - test-echelon.C, 803
- test\_rowechelon
  - test-echelon.C, 803
- test\_RPM\_fsytrf
  - test-fsytrf.C, 821
- test\_square\_fgesv
  - test-fgesv.C, 814
- testLTQSRPM
  - test-quasisep.C, 842
- TestOneMethod
  - TestOneMethod< Simd >, 509
- TestOneMethod< Simd >, 508
  - Element, 509
  - enable\_if\_t, 509
  - evaluate\_scalar\_method, 509, 510
  - evaluate\_simd\_method, 510

- getStatus, [510](#)
- getTestName, [510](#)
- inputs, [511](#)
- name, [511](#)
- nb\_lref, [510](#)
- outputs\_scalar, [511](#)
- outputs\_simd, [511](#)
- TestOneMethod, [509](#)
- vect\_size, [510](#)
- vect\_t, [509](#)
- vectElt, [509](#)
- writeDebugData, [510](#)
- writeResultLine, [510](#)
- tfn\_minus, [511](#)
  - operator(), [511](#)
- tfn\_minus\_eq, [511](#)
  - operator(), [512](#)
- tfn\_mul, [512](#)
  - operator(), [512](#)
- tfn\_mul\_eq, [512](#)
  - operator(), [512](#)
- tfn\_plus, [512](#)
  - operator(), [513](#)
- tfn\_plus\_eq, [513](#)
  - operator(), [513](#)
- tgemm
  - test-lu.C, [836](#)
  - test-permutations.C, [841](#)
- THREAD\_INDEX
  - parallel.h, [777](#)
- THREADS
  - benchmark-fgemm-rns.C, [532](#)
- Threads, [513](#)
- threads\_fgemm
  - FFPACK, [361](#)
- threads\_ftsm
  - FFPACK, [361](#)
- THREED
  - benchmark-fgemm-rns.C, [532](#)
- ThreeD, [513](#)
- THREEDA
  - benchmark-fgemm-rns.C, [532](#)
- ThreeDAdaptive, [513](#)
- ThreeDInPlace, [514](#)
- THREEDIP
  - benchmark-fgemm-rns.C, [532](#)
- TIME
  - test-fger.C, [812](#)
- TIME\_CHECKER\_CHARPOLY
  - test-charpoly-check.C, [798](#)
- TIME\_CHECKER\_Det
  - test-det-check.C, [801](#)
- Timer
  - FFLAS, [74](#)
- timer.h, [794](#)
- timtot
  - test-lu.C, [836](#)
  - test-permutations.C, [841](#)
- TInverter
  - FFPACK, [344](#), [347](#)
- tmain
  - benchmark-fgemm-mp.C, [531](#)
  - benchmark-fgemv-mp.C, [534](#)
- Todo List, [9](#)
- tperm
  - test-lu.C, [836](#)
  - test-permutations.C, [841](#)
- transpose
  - BlockTransposeSIMD< Field, Simd, >, [403](#), [404](#)
- trest
  - test-lu.C, [836](#)
  - test-permutations.C, [841](#)
- trinv\_left
  - FFPACK, [309](#), [365](#)
- trinv\_left\_modular\_double
  - ffpack.C, [735](#)
  - ffpack\_c.h, [757](#)
- TRSMBound
  - FFLAS::Protected, [212](#)
- TRSMHelper
  - TRSMHelper< ReclterTrait, ParSeqTrait >, [514](#)
- TRSMHelper< ReclterTrait, ParSeqTrait >, [514](#)
  - parseq, [515](#)
  - pMMH, [515](#)
  - TRSMHelper, [514](#)
- TTimer
  - arithprog.C, [517](#)
  - benchmark-dgemm.C, [526](#)
  - charpoly.C, [518](#)
- ttrsm
  - test-lu.C, [836](#)
  - test-permutations.C, [841](#)
- Tutorial, [2](#)
- TWOD
  - benchmark-fgemm-rns.C, [532](#)
- TwoD, [515](#)
- TWODA
  - benchmark-fgemm-rns.C, [532](#)
- TwoDAdaptive, [515](#)
- type
  - Argument, [398](#)
  - associatedDelayedField< Field >, [399](#)
  - CompactElement< Element >, [412](#)
  - is\_simd< T >, [449](#)
- TYPE\_BOOL
  - args-parser.h, [785](#)
- TYPE\_DOUBLE
  - args-parser.h, [786](#)
- TYPE\_INT
  - args-parser.h, [786](#)
- TYPE\_INTEGER
  - args-parser.h, [786](#)
- type\_integer
  - args-parser.h, [785](#)
- TYPE\_INTLIST
  - args-parser.h, [786](#)

- TYPE\_LONGLONG
  - args-parser.h, 786
- TYPE\_NONE
  - args-parser.h, 785
- TYPE\_STR
  - args-parser.h, 786
- type\_string
  - NoSimd< T >, 458
- TYPE\_UINT64
  - args-parser.h, 786
- unfit
  - FFLAS::Protected, 220, 221
- unpackhi
  - ScalFunctions< Element >, 494
- unpacklo
  - ScalFunctions< Element >, 494
- unpacklohi
  - ScalFunctions< Element >, 495
- UnparametricTag, 515
- updateD
  - FFPACK::Protected, 394
- USE\_OPENMP
  - config.h, 564
- UserTimer
  - FFLAS, 74
- utils.h, 657
- val
  - Coo< Field >, 417
  - Coo< ValT, IdxT >, 415, 418
- valid
  - NoSimd< T >, 458
- VALUE
  - parallel.h, 778
- value
  - AlgoChooser< ModeT, ParSeq >, 397
  - AreEqual< X, Y >, 398
  - compatible\_data\_type< Field >, 412
  - ElementTraits< Element >, 421
  - has\_minus\_eq\_impl< C >, 443
  - has\_minus\_impl< C >, 443
  - has\_mul\_eq\_impl< C >, 444
  - has\_mul\_impl< C >, 444
  - has\_operation< T >, 444
  - has\_plus\_eq\_impl< C >, 445
  - has\_plus\_impl< C >, 445
  - is\_simd< T >, 449
  - ModeTraits< Field >, 456
  - need\_field\_characteristic< Field >, 457
  - width< T >, 516
- vand
  - ScalFunctions< Element >, 492
  - Simd128i\_base, 497
  - Simd512i\_base, 500
- vandnot
  - ScalFunctions< Element >, 492
  - Simd128i\_base, 498
  - Simd512i\_base, 500
- VEC\_ADD
  - FFLAS::vectorised, 279
- VEC\_SUB
  - FFLAS::vectorised, 279
- vect\_size
  - FieldSimd< \_Field >, 430
  - NoSimd< T >, 459
  - TestOneMethod< Simd >, 510
- vect\_t
  - FieldSimd< \_Field >, 426
  - NoSimd< T >, 458
  - simd128\_int64.inl, 620
  - Simd128i\_base, 497
  - simd256\_int64.inl, 623
  - Simd256i\_base, 499
  - simd512\_int64.inl, 625
  - Simd512i\_base, 499
  - TestOneMethod< Simd >, 509
- vectElt
  - ScalFunctions< Element >, 491
  - TestOneMethod< Simd >, 509
- vector< T >, 516
  - elements, 516
- vector< T >::const\_iterator, 413
- vector< T >::const\_reverse\_iterator, 414
- vector< T >::iterator, 450
- vector< T >::reverse\_iterator, 463
- verification\_PLUQ
  - benchmark-pluq.C, 545
- verifPLUQ
  - test-lu.C, 834
- VERSION
  - config.h, 564
- vor
  - ScalFunctions< Element >, 492
  - Simd128i\_base, 498
  - Simd512i\_base, 500
- vxor
  - ScalFunctions< Element >, 492
  - Simd128i\_base, 498
  - Simd512i\_base, 500
- WAIT
  - parallel.h, 777
- width< T >, 516
  - value, 516
- Winograd, 516
  - FFLAS::BLAS3, 193
- winograd.C, 522
  - balanced, 523
  - DOUBLE\_TO\_FLOAT\_CROSSOVER, 522
  - GFOPS, 522
  - main, 523
- 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, [217](#)  
 WinogradPar, [516](#)  
 WinogradSteps  
     FFLAS::Protected, [216](#)  
 WinogradThreshold  
     FFLAS::Protected, [215](#)  
 WinoPar  
     FFLAS::BLAS3, [193](#)  
 WINOTHRESHOLD  
     fflas.h, [570](#)  
 WRITE  
     parallel.h, [778](#)  
 write  
     RNSInteger< RNS >, [482](#)  
     RNSIntegerMod< RNS >, [487](#)  
 write\_field  
     Matio.h, [793](#)  
 write\_matrix  
     benchmark-fgemv-mp.C, [534](#)  
     RNSIntegerMod< RNS >, [487](#)  
 write\_matrix\_long  
     RNSIntegerMod< RNS >, [488](#)  
 writeCommandString  
     FFLAS, [185](#)  
 writeDebugData  
     TestOneMethod< Simd >, [510](#)  
 writeDnsFormat  
     FFLAS, [152](#)  
 WriteMatrix  
     FFLAS, [185](#), [187](#)  
 WritePermutation  
     FFLAS, [187](#)  
 writeResultLine  
     TestOneMethod< Simd >, [510](#)  
 zero  
     FFLAS, [76](#)  
     FieldSimd< \_Field >, [428](#)  
     RNSInteger< RNS >, [482](#)  
     RNSIntegerMod< RNS >, [489](#)  
     ScalFunctions< Element >, [492](#)  
     Simd128i\_base, [497](#)  
     Simd256i\_base, [499](#)  
     Simd512i\_base, [500](#)  
 ZOSparseMatrix  
     FFLAS, [72](#)