Fast Auxiliary Space Preconditioning

1.8.8 Dec/30/2016

Generated by Doxygen 1.8.12

Contents

1 Introduction **1**

2 How to obtain FASP **3**

3 Building and Installation **5**

4 Developers **7**

5 Doxygen **9**

6 Todo List **11**

7 Data Structure Index **13**

7.1 Data Structures 13

8 File Index **17**

8.1 File List 17

9 Data Structure Documentation **23**

9.1 AMG_data Struct Reference 23

9.1.1 Detailed Description 24

9.2 AMG_data_bsr Struct Reference 24

9.2.1 Detailed Description 26

9.3 AMG_param Struct Reference 26

9.3.1 Detailed Description 28

9.4 block_BSR Struct Reference 28
9.4.1 Detailed Description 29
9.5 block_dvector Struct Reference 29
9.5.1 Detailed Description 29
9.6 block_iCSRmat Struct Reference 30
9.6.1 Detailed Description 30
9.7 block_ivector Struct Reference 30
9.7.1 Detailed Description 31
9.8 block_Reservoir Struct Reference 31
9.8.1 Detailed Description 31
9.9 dBLCmat Struct Reference 32
9.9.1 Detailed Description 32
9.10 dBSRmat Struct Reference 32
9.10.1 Detailed Description 33
9.10.2 Field Documentation 33
9.10.2.1 JA 33
9.10.2.2 val 33
9.11 dCOOmat Struct Reference 33
9.11.1 Detailed Description 34
9.12 dCSRLmat Struct Reference 34
9.12.1 Detailed Description 35
9.13 dCSRmat Struct Reference 35
9.13.1 Detailed Description 36
9.14 ddenmat Struct Reference 36
9.14.1 Detailed Description 36
9.15 dSTRmat Struct Reference 37
9.15.1 Detailed Description 37
9.16 dvector Struct Reference 38

9.16.1 Detailed Description 38
9.17 grid2d Struct Reference 38
9.17.1 Detailed Description 39
9.17.2 Field Documentation 39
9.17.2.1 e 39
9.17.2.2 edges 39
9.17.2.3 ediri 39
9.17.2.4 efather 39
9.17.2.5 p 40
9.17.2.6 pdiri 40
9.17.2.7 pfather 40
9.17.2.8 s 40
9.17.2.9 t 40
9.17.2.10 tfather 40
9.17.2.11 triangles 41
9.17.2.12 vertices 41
9.18 iCOOmat Struct Reference 41
9.18.1 Detailed Description 42
9.19 iCSRmat Struct Reference 42
9.19.1 Detailed Description 42
9.20 idenmat Struct Reference 43
9.20.1 Detailed Description 43
9.21 ILU_data Struct Reference 43
9.21.1 Detailed Description 44
9.22 ILU_param Struct Reference 45
9.22.1 Detailed Description 45
9.23 input_param Struct Reference 45
9.23.1 Detailed Description 47

9.23.2 Field Documentation . . . 47
9.23.2.1 AMG_aggregation_type . . . 47
9.23.2.2 AMG_aggressive_level . . . 47
9.23.2.3 AMG_aggressive_path . . . 47
9.23.2.4 AMG_amli_degree . . . 47
9.23.2.5 AMG_coarse_dof . . . 48
9.23.2.6 AMG_coarse_scaling . . . 48
9.23.2.7 AMG_coarse_solver . . . 48
9.23.2.8 AMG_coarsening_type . . . 48
9.23.2.9 AMG_cycle_type . . . 48
9.23.2.10 AMG_ILU_levels . . . 48
9.23.2.11 AMG_interpolation_type . . . 49
9.23.2.12 AMG_levels . . . 49
9.23.2.13 AMG_max_aggregation . . . 49
9.23.2.14 AMG_max_row_sum . . . 49
9.23.2.15 AMG_maxit . . . 49
9.23.2.16 AMG_nl_amli_krylov_type . . . 49
9.23.2.17 AMG_pair_number . . . 50
9.23.2.18 AMG_polynomial_degree . . . 50
9.23.2.19 AMG_postsmooth_iter . . . 50
9.23.2.20 AMG_presmooth_iter . . . 50
9.23.2.21 AMG_quality_bound . . . 50
9.23.2.22 AMG_relaxation . . . 50
9.23.2.23 AMG_Schwarz_levels . . . 51
9.23.2.24 AMG_smooth_filter . . . 51
9.23.2.25 AMG_smooth_order . . . 51
9.23.2.26 AMG_smoother . . . 51
9.23.2.27 AMG_strong_coupled . . . 51

9.23.2.28 AMG_strong_threshold 51
9.23.2.29 AMG_tentative_smooth 52
9.23.2.30 AMG_tol 52
9.23.2.31 AMG_truncation_threshold 52
9.23.2.32 AMG_type 52
9.23.2.33 ILU_droptol 52
9.23.2.34 ILU_lfil 52
9.23.2.35 ILU_permtol 53
9.23.2.36 ILU_relax 53
9.23.2.37 ILU_type 53
9.23.2.38 inifile 53
9.23.2.39 itsolver_maxit 53
9.23.2.40 itsolver_tol 53
9.23.2.41 output_type 54
9.23.2.42 precond_type 54
9.23.2.43 print_level 54
9.23.2.44 problem_num 54
9.23.2.45 restart 54
9.23.2.46 Schwarz_blksolver 54
9.23.2.47 Schwarz_maxlvl 55
9.23.2.48 Schwarz_mmsize 55
9.23.2.49 Schwarz_type 55
9.23.2.50 solver_type 55
9.23.2.51 stop_type 55
9.23.2.52 workdir 56
9.24 itsolver_param Struct Reference 56
9.24.1 Detailed Description 56
9.24.2 Field Documentation 56

9.24.2.1 itsolver_type 56
9.24.2.2 maxit 57
9.24.2.3 precond_type 57
9.24.2.4 print_level 57
9.24.2.5 restart 57
9.24.2.6 stop_type 57
9.24.2.7 tol 58
9.25 ivector Struct Reference 58
9.25.1 Detailed Description 58
9.26 Link Struct Reference 58
9.26.1 Detailed Description 59
9.27 linked_list Struct Reference 59
9.27.1 Detailed Description 59
9.28 mallinfo Struct Reference 60
9.28.1 Detailed Description 60
9.29 malloc_chunk Struct Reference 60
9.29.1 Detailed Description 60
9.30 malloc_params Struct Reference 61
9.30.1 Detailed Description 61
9.31 malloc_segment Struct Reference 61
9.31.1 Detailed Description 61
9.32 malloc_state Struct Reference 62
9.32.1 Detailed Description 62
9.33 malloc_tree_chunk Struct Reference 62
9.33.1 Detailed Description 63
9.34 Mumps_data Struct Reference 63
9.34.1 Detailed Description 63
9.35 mxv_matfree Struct Reference 63

9.35.1 Detailed Description . . . 64
9.36 nedmallinfo Struct Reference . . . 64
9.36.1 Detailed Description . . . 64
9.37 Pardiso_data Struct Reference . . . 64
9.37.1 Detailed Description . . . 65
9.38 precond Struct Reference . . . 65
9.38.1 Detailed Description . . . 65
9.39 precond_block_data Struct Reference . . . 65
9.39.1 Detailed Description . . . 66
9.39.2 Field Documentation . . . 66
9.39.2.1 A_diag . . . 66
9.39.2.2 Ablc . . . 66
9.39.2.3 amgparam . . . 66
9.39.2.4 LU_diag . . . 67
9.39.2.5 mgl . . . 67
9.39.2.6 r . . . 67
9.40 precond_block_reservoir_data Struct Reference . . . 67
9.40.1 Detailed Description . . . 69
9.40.2 Field Documentation . . . 69
9.40.2.1 diag . . . 69
9.40.2.2 diaginv . . . 69
9.40.2.3 diaginvS . . . 70
9.40.2.4 order . . . 70
9.40.2.5 perf_idx . . . 70
9.40.2.6 pivot . . . 70
9.40.2.7 pivotS . . . 70
9.40.2.8 PP . . . 70
9.40.2.9 r . . . 71

9.40.2.10 RR . . . 71
9.40.2.11 scaled . . . 71
9.40.2.12 SS . . . 71
9.40.2.13 w . . . 71
9.40.2.14 WW . . . 72
9.41 precond_data Struct Reference . . . 72
9.41.1 Detailed Description . . . 73
9.42 precond_data_bsr Struct Reference . . . 73
9.42.1 Detailed Description . . . 75
9.43 precond_data_str Struct Reference . . . 75
9.43.1 Detailed Description . . . 76
9.44 precond_diagbsr Struct Reference . . . 77
9.44.1 Detailed Description . . . 77
9.45 precond_diagstr Struct Reference . . . 77
9.45.1 Detailed Description . . . 78
9.46 precond_FASP_blkoil_data Struct Reference . . . 78
9.46.1 Detailed Description . . . 79
9.46.2 Field Documentation . . . 80
9.46.2.1 A . . . 80
9.46.2.2 diaginv . . . 80
9.46.2.3 diaginv_noscale . . . 80
9.46.2.4 diaginv_S . . . 80
9.46.2.5 LU_P . . . 80
9.46.2.6 LU_S . . . 81
9.46.2.7 maxit . . . 81
9.46.2.8 mgl_data . . . 81
9.46.2.9 neigh . . . 81
9.46.2.10 order . . . 81

9.46.2.11 perf_idx 81
9.46.2.12 perf_neigh 82
9.46.2.13 pivot 82
9.46.2.14 pivot_S 82
9.46.2.15 PP 82
9.46.2.16 r 82
9.46.2.17 restart 82
9.46.2.18 RR 83
9.46.2.19 scaled 83
9.46.2.20 SS 83
9.46.2.21 tol 83
9.46.2.22 w 83
9.46.2.23 WW 84
9.47 precond_sweeping_data Struct Reference 84
9.47.1 Detailed Description 84
9.47.2 Field Documentation 85
9.47.2.1 A 85
9.47.2.2 Ai 85
9.47.2.3 local_A 85
9.47.2.4 local_index 85
9.47.2.5 local_LU 85
9.47.2.6 NumLayers 86
9.47.2.7 r 86
9.47.2.8 w 86
9.48 Schwarz_data Struct Reference 86
9.48.1 Detailed Description 87
9.49 Schwarz_param Struct Reference 88
9.49.1 Detailed Description 88

10 File Documentation 89
10.1 amg.c File Reference 89
10.1.1 Detailed Description 89
10.1.2 Function Documentation 89
10.1.2.1 fasp_solver_amg() 89
10.2 amg_setup_cr.c File Reference 90
10.2.1 Detailed Description 91
10.2.2 Function Documentation 91
10.2.2.1 fasp_amg_setup_cr() 91
10.3 amg_setup_rs.c File Reference 92
10.3.1 Detailed Description 92
10.3.2 Function Documentation 92
10.3.2.1 fasp_amg_setup_rs() 92
10.4 amg_setup_sa.c File Reference 93
10.4.1 Detailed Description 93
10.4.2 Function Documentation 93
10.4.2.1 fasp_amg_setup_sa() 93
10.4.2.2 fasp_amg_setup_sa_bsr() 94
10.5 amg_setup_ua.c File Reference 95
10.5.1 Detailed Description 95
10.5.2 Function Documentation 95
10.5.2.1 fasp_amg_setup_ua() 95
10.5.2.2 fasp_amg_setup_ua_bsr() 96
10.6 amg_solve.c File Reference 96
10.6.1 Detailed Description 97
10.6.2 Function Documentation 97
10.6.2.1 fasp_amg_solve() 97
10.6.2.2 fasp_amg_solve_amli() 98

10.6.2.3 fasp_amg_solve_nl_amli() 98
10.6.2.4 fasp_famg_solve() 99
10.7 amlirecur.c File Reference 100
10.7.1 Detailed Description 100
10.7.2 Function Documentation 100
10.7.2.1 fasp_amg_amli_coef() 100
10.7.2.2 fasp_solver_amli() 101
10.7.2.3 fasp_solver_nl_amli() 102
10.7.2.4 fasp_solver_nl_amli_bsr() 102
10.8 array.c File Reference 103
10.8.1 Detailed Description 104
10.8.2 Function Documentation 104
10.8.2.1 fasp_array_cp() 104
10.8.2.2 fasp_array_cp_nc3() 104
10.8.2.3 fasp_array_cp_nc5() 105
10.8.2.4 fasp_array_cp_nc7() 105
10.8.2.5 fasp_array_invpermut_nb() 106
10.8.2.6 fasp_array_null() 107
10.8.2.7 fasp_array_permut_nb() 107
10.8.2.8 fasp_array_set() 108
10.8.2.9 fasp_iarray_cp() 108
10.8.2.10 fasp_iarray_set() 109
10.9 blas_array.c File Reference 109
10.9.1 Detailed Description 110
10.9.2 Function Documentation 110
10.9.2.1 fasp_blas_array_ax() 110
10.9.2.2 fasp_blas_array_axpby() 111
10.9.2.3 fasp_blas_array_axpy() 112

10.9.2.4 fasp_blas_array_axpyz() 112

10.9.2.5 fasp_blas_array_dotprod() 113

10.9.2.6 fasp_blas_array_norm1() 113

10.9.2.7 fasp_blas_array_norm2() 114

10.9.2.8 fasp_blas_array_norminf() 115

10.10blas_blc.c File Reference 115

10.10.1 Detailed Description 116

10.10.2 Function Documentation 116

10.10.2.1 fasp_blas_bdbsr_aAxpy() 116

10.10.2.2 fasp_blas_bdbsr_mxv() 116

10.10.2.3 fasp_blas_dblc_aAxpy() 117

10.10.2.4 fasp_blas_dblc_mxv() 117

10.11blas_bsr.c File Reference 118

10.11.1 Detailed Description 119

10.11.2 Function Documentation 119

10.11.2.1 fasp_blas_dbsr_aAxpby() 119

10.11.2.2 fasp_blas_dbsr_aAxpy() 119

10.11.2.3 fasp_blas_dbsr_aAxpy_agg() 120

10.11.2.4 fasp_blas_dbsr_axm() 121

10.11.2.5 fasp_blas_dbsr_mxm() 121

10.11.2.6 fasp_blas_dbsr_mxv() 122

10.11.2.7 fasp_blas_dbsr_mxv_agg() 123

10.11.2.8 fasp_blas_dbsr_rap() 123

10.11.2.9 fasp_blas_dbsr_rap1() 124

10.11.2.10fasp_blas_dbsr_rap_agg() 125

10.12blas_csr.c File Reference 125

10.12.1 Detailed Description 126

10.12.2 Function Documentation 127

10.12.2.1 fasp_blas_dcsr_aAxpy() . . . 127
10.12.2.2 fasp_blas_dcsr_aAxpy_agg() . . . 127
10.12.2.3 fasp_blas_dcsr_add() . . . 128
10.12.2.4 fasp_blas_dcsr_axm() . . . 129
10.12.2.5 fasp_blas_dcsr_bandwith() . . . 129
10.12.2.6 fasp_blas_dcsr_mxm() . . . 130
10.12.2.7 fasp_blas_dcsr_mxv() . . . 130
10.12.2.8 fasp_blas_dcsr_mxv_agg() . . . 131
10.12.2.9 fasp_blas_dcsr_ptap() . . . 131
10.12.2.10 fasp_blas_dcsr_rap() . . . 132
10.12.2.11 fasp_blas_dcsr_rap4() . . . 133
10.12.2.12 fasp_blas_dcsr_rap_agg() . . . 134
10.12.2.13 fasp_blas_dcsr_rap_agg1() . . . 134
10.12.2.14 fasp_blas_dcsr_vmv() . . . 135
10.13 blas_csrl.c File Reference . . . 135
10.13.1 Detailed Description . . . 136
10.13.2 Function Documentation . . . 136
10.13.2.1 fasp_blas_dcsrl_mxv() . . . 136
10.14 blas_smat.c File Reference . . . 136
10.14.1 Detailed Description . . . 138
10.14.2 Function Documentation . . . 138
10.14.2.1 fasp_blas_array_axpy_nc2() . . . 138
10.14.2.2 fasp_blas_array_axpy_nc3() . . . 139
10.14.2.3 fasp_blas_array_axpy_nc5() . . . 139
10.14.2.4 fasp_blas_array_axpy_nc7() . . . 141
10.14.2.5 fasp_blas_array_axpyz_nc2() . . . 141
10.14.2.6 fasp_blas_array_axpyz_nc3() . . . 142
10.14.2.7 fasp_blas_array_axpyz_nc5() . . . 143

10.14.2.8 fasp_blas_array_axpyz_nc7() 144
10.14.2.9 fasp_blas_smat_aAxpby() 144
10.14.2.10 fasp_blas_smat_add() 145
10.14.2.11 fasp_blas_smat_axm() 145
10.14.2.12 fasp_blas_smat_mul() 146
10.14.2.13 fasp_blas_smat_mul_nc2() 147
10.14.2.14 fasp_blas_smat_mul_nc3() 147
10.14.2.15 fasp_blas_smat_mul_nc5() 148
10.14.2.16 fasp_blas_smat_mul_nc7() 148
10.14.2.17 fasp_blas_smat_mxv() 149
10.14.2.18 fasp_blas_smat_mxv_nc2() 149
10.14.2.19 fasp_blas_smat_mxv_nc3() 150
10.14.2.20 fasp_blas_smat_mxv_nc5() 150
10.14.2.21 fasp_blas_smat_mxv_nc7() 152
10.14.2.22 fasp_blas_smat_ymAx() 153
10.14.2.23 fasp_blas_smat_ymAx_nc2() 153
10.14.2.24 fasp_blas_smat_ymAx_nc3() 154
10.14.2.25 fasp_blas_smat_ymAx_nc5() 154
10.14.2.26 fasp_blas_smat_ymAx_nc7() 155
10.14.2.27 fasp_blas_smat_ymAx_ns() 156
10.14.2.28 fasp_blas_smat_ymAx_ns2() 156
10.14.2.29 fasp_blas_smat_ymAx_ns3() 157
10.14.2.30 fasp_blas_smat_ymAx_ns5() 158
10.14.2.31 fasp_blas_smat_ymAx_ns7() 158
10.14.2.32 fasp_blas_smat_ypAx() 159
10.14.2.33 fasp_blas_smat_ypAx_nc2() 159
10.14.2.34 fasp_blas_smat_ypAx_nc3() 160
10.14.2.35 fasp_blas_smat_ypAx_nc5() 160

10.14.2.36 fasp_blas_smat_ypAx_nc7() 162
10.15 blas_str.c File Reference 162
10.15.1 Detailed Description 163
10.15.2 Function Documentation 163
10.15.2.1 fasp_blas_dstr_aAxpy() 163
10.15.2.2 fasp_blas_dstr_mxv() 164
10.15.2.3 fasp_dstr_diagscale() 164
10.16 blas_vec.c File Reference 165
10.16.1 Detailed Description 165
10.16.2 Function Documentation 165
10.16.2.1 fasp_blas_dvec_axpy() 165
10.16.2.2 fasp_blas_dvec_axpyz() 166
10.16.2.3 fasp_blas_dvec_dotprod() 167
10.16.2.4 fasp_blas_dvec_norm1() 167
10.16.2.5 fasp_blas_dvec_norm2() 168
10.16.2.6 fasp_blas_dvec_norminf() 169
10.16.2.7 fasp_blas_dvec_relerr() 169
10.17 checkmat.c File Reference 170
10.17.1 Detailed Description 170
10.17.2 Function Documentation 170
10.17.2.1 fasp_check_dCSRmat() 170
10.17.2.2 fasp_check_diagdom() 171
10.17.2.3 fasp_check_diagpos() 172
10.17.2.4 fasp_check_diagzero() 172
10.17.2.5 fasp_check_iCSRmat() 173
10.17.2.6 fasp_check_symm() 173
10.18 coarsening_cr.c File Reference 174
10.18.1 Detailed Description 174

10.18.2 Function Documentation 174
10.18.2.1 fasp_amg_coarsening_cr() 174
10.19coarsening_rs.c File Reference 175
10.19.1 Detailed Description 175
10.19.2 Function Documentation 175
10.19.2.1 fasp_amg_coarsening_rs() 175
10.20convert.c File Reference 176
10.20.1 Detailed Description 177
10.20.2 Function Documentation 177
10.20.2.1 endian_convert_int() 177
10.20.2.2 endian_convert_real() 177
10.20.2.3 fasp_aux_bbyteToldouble() 178
10.20.2.4 fasp_aux_change_endian4() 179
10.20.2.5 fasp_aux_change_endian8() 179
10.21doxygen.h File Reference 180
10.21.1 Detailed Description 180
10.22eigen.c File Reference 180
10.22.1 Detailed Description 180
10.22.2 Function Documentation 180
10.22.2.1 fasp_dcsr_eig() 180
10.23famg.c File Reference 181
10.23.1 Detailed Description 181
10.23.2 Function Documentation 181
10.23.2.1 fasp_solver_famg() 181
10.24fasp.h File Reference 182
10.24.1 Detailed Description 185
10.24.2 Macro Definition Documentation 185
10.24.2.1 __FASP_HEADER__ 185

10.24.2.2 ABS . . . 185
10.24.2.3 DIAGONAL_PREF . . . 186
10.24.2.4 DLMALLOC . . . 186
10.24.2.5 FASP_GSRB . . . 186
10.24.2.6 FASP_USE_ILU . . . 186
10.24.2.7 FASP_VERSION . . . 186
10.24.2.8 GE . . . 187
10.24.2.9 GT . . . 187
10.24.2.10 ILU_C_VERSION . . . 187
10.24.2.11 INT . . . 187
10.24.2.12 ISNAN . . . 187
10.24.2.13 LE . . . 188
10.24.2.14 LONG . . . 188
10.24.2.15 LONGLONG . . . 188
10.24.2.16 LS . . . 188
10.24.2.17 MAX . . . 188
10.24.2.18 MIN . . . 189
10.24.2.19 NEDMALLOC . . . 189
10.24.2.20 PUT_INT . . . 189
10.24.2.21 PUT_REAL . . . 189
10.24.2.22 REAL . . . 189
10.24.2.23 RS_C1 . . . 190
10.24.2.24 SHORT . . . 190
10.24.3 Typedef Documentation . . . 190
10.24.3.1 dCOOmat . . . 190
10.24.3.2 dCSRLmat . . . 190
10.24.3.3 dCSRmat . . . 190
10.24.3.4 ddenmat . . . 191

10.24.3.5 dSTRmat . . . 191
10.24.3.6 dvector . . . 191
10.24.3.7 grid2d . . . 191
10.24.3.8 iCOOmat . . . 191
10.24.3.9 iCSRmat . . . 191
10.24.3.10 idenmat . . . 191
10.24.3.11 ivector . . . 192
10.24.3.12 LinkList . . . 192
10.24.3.13 ListElement . . . 192
10.24.3.14 pcgrid2d . . . 192
10.24.3.15 pgrid2d . . . 192
10.24.4 Variable Documentation . . . 192
10.24.4.1 count . . . 192
10.24.4.2 IMAP . . . 193
10.24.4.3 MAXIMAP . . . 193
10.24.4.4 nx_rb . . . 193
10.24.4.5 ny_rb . . . 193
10.24.4.6 nz_rb . . . 193
10.24.4.7 total_alloc_count . . . 193
10.24.4.8 total_alloc_mem . . . 194
10.25 fasp_block.h File Reference . . . 194
10.25.1 Detailed Description . . . 195
10.25.2 Macro Definition Documentation . . . 195
10.25.2.1 __FASPBLOCK_HEADER__ . . . 195
10.25.2.2 SMOOTHER_BLKOIL . . . 195
10.25.2.3 SMOOTHER_SPETEN . . . 196
10.25.3 Typedef Documentation . . . 196
10.25.3.1 block_BSR . . . 196

10.25.3.2 block_dvector 196
10.25.3.3 block_iCSRmat 196
10.25.3.4 block_ivector 196
10.25.3.5 block_Reservoir 196
10.25.3.6 dBLCmat 197
10.25.3.7 dBSRmat 197
10.25.3.8 precond_block_reservoir_data 197
10.26 fasp_const.h File Reference 197
10.26.1 Detailed Description 200
10.26.2 Macro Definition Documentation 201
10.26.2.1 AMLI_CYCLE 201
10.26.2.2 ASCEND 201
10.26.2.3 BIGREAL 201
10.26.2.4 CF_ORDER 201
10.26.2.5 CGPT 201
10.26.2.6 CLASSIC_AMG 202
10.26.2.7 COARSE_AC 202
10.26.2.8 COARSE_CR 202
10.26.2.9 COARSE_MIS 202
10.26.2.10 COARSE_RS 202
10.26.2.11 COARSE_RSP 203
10.26.2.12 CPFIRST 203
10.26.2.13 DESCEND 203
10.26.2.14 ERROR_ALLOC_MEM 203
10.26.2.15 ERROR_AMG_COARSE_TYPE 203
10.26.2.16 ERROR_AMG_COARSEING 203
10.26.2.17 ERROR_AMG_INTERP_TYPE 204
10.26.2.18 ERROR_AMG_SMOOTH_TYPE 204

10.26.2.19 ERROR_DATA_STRUCTURE 204
10.26.2.20 ERROR_DATA_ZERODIAG 204
10.26.2.21 ERROR_DUMMY_VAR 204
10.26.2.22 ERROR_INPUT_PAR 204
10.26.2.23 ERROR_LIC_TYPE 205
10.26.2.24 ERROR_MAT_SIZE 205
10.26.2.25 ERROR_MISC 205
10.26.2.26 ERROR_NUM_BLOCKS 205
10.26.2.27 ERROR_OPEN_FILE 205
10.26.2.28 ERROR_QUAD_DIM 205
10.26.2.29 ERROR_QUAD_TYPE 206
10.26.2.30 ERROR_REGRESS 206
10.26.2.31 ERROR_SOLVER_EXIT 206
10.26.2.32 ERROR_SOLVER_ILUSETUP 206
10.26.2.33 ERROR_SOLVER_MAXIT 206
10.26.2.34 ERROR_SOLVER_MISC 206
10.26.2.35 ERROR_SOLVER_PRECTYPE 207
10.26.2.36 ERROR_SOLVER_SOLSTAG 207
10.26.2.37 ERROR_SOLVER_STAG 207
10.26.2.38 ERROR_SOLVER_TOLSMALL 207
10.26.2.39 ERROR_SOLVER_TYPE 207
10.26.2.40 ERROR_UNKNOWN 207
10.26.2.41 ERROR_WRONG_FILE 208
10.26.2.42 FALSE 208
10.26.2.43 FASP_SUCCESS 208
10.26.2.44 FGPT 208
10.26.2.45 FPFIRST 208
10.26.2.46 G0PT 209

10.26.2.47 ILU_MC_OMP 209
10.26.2.48 ILUk 209
10.26.2.49 ILUt 209
10.26.2.50 ILUtp 209
10.26.2.51 INTERP_DIR 210
10.26.2.52 INTERP_ENG 210
10.26.2.53 INTERP_STD 210
10.26.2.54 ISPT 210
10.26.2.55 MAT_bBSR 210
10.26.2.56 MAT_BLC 211
10.26.2.57 MAT_BSR 211
10.26.2.58 MAT_CSR 211
10.26.2.59 MAT_CSRL 211
10.26.2.60 MAT_FREE 211
10.26.2.61 MAT_STR 212
10.26.2.62 MAT_SymCSR 212
10.26.2.63 MAX_AMG_LVL 212
10.26.2.64 MAX_CRATE 212
10.26.2.65 MAX_REFINE_LVL 212
10.26.2.66 MAX_RESTART 212
10.26.2.67 MAX_STAG 213
10.26.2.68 MIN_CDOF 213
10.26.2.69 MIN_CRATE 213
10.26.2.70 NL_AMLI_CYCLE 213
10.26.2.71 NO_ORDER 213
10.26.2.72 OFF 214
10.26.2.73 ON 214
10.26.2.74 OPENMP_HOLDS 214

10.26.2.75 PAIRWISE 214

10.26.2.76 PREC_AMG 214

10.26.2.77 PREC_DIAG 215

10.26.2.78 PREC_FMG 215

10.26.2.79 PREC_ILU 215

10.26.2.80 PREC_NULL 215

10.26.2.81 PREC_SCHWARZ 215

10.26.2.82 PRINT_ALL 216

10.26.2.83 PRINT_MIN 216

10.26.2.84 PRINT_MORE 216

10.26.2.85 PRINT_MOST 216

10.26.2.86 PRINT_NONE 216

10.26.2.87 PRINT_SOME 217

10.26.2.88 SA_AMG 217

10.26.2.89 SCHWARZ_BACKWARD 217

10.26.2.90 SCHWARZ_FORWARD 217

10.26.2.91 SCHWARZ_SYMMETRIC 217

10.26.2.92 SMALLREAL 218

10.26.2.93 SMALLREAL2 218

10.26.2.94 SMOOTHER_CG 218

10.26.2.95 SMOOTHER_GS 218

10.26.2.96 SMOOTHER_GSOR 218

10.26.2.97 SMOOTHER_JACOBI 218

10.26.2.98 SMOOTHER_L1DIAG 219

10.26.2.99 SMOOTHER_POLY 219

10.26.2.100 SMOOTHER_SGS 219

10.26.2.101 SMOOTHER_SGSOR 219

10.26.2.102 SMOOTHER_SOR 219

10.26.2.103 SMOOTHER_SSOR 219
10.26.2.104 SOLVER_AMG 220
10.26.2.105 SOLVER_BiCGstab 220
10.26.2.106 SOLVER_CG 220
10.26.2.107 SOLVER_DEFAULT 220
10.26.2.108 SOLVER_FMG 220
10.26.2.109 SOLVER_GCG 221
10.26.2.110 SOLVER_GCR 221
10.26.2.111 SOLVER_GMRES 221
10.26.2.112 SOLVER_MinRes 221
10.26.2.113 SOLVER_MUMPS 221
10.26.2.114 SOLVER_PARDISO 221
10.26.2.115 SOLVER_SBiCGstab 222
10.26.2.116 SOLVER_SCG 222
10.26.2.117 SOLVER_SGCG 222
10.26.2.118 SOLVER_SGMRES 222
10.26.2.119 SOLVER_SMinRes 222
10.26.2.120 SOLVER_SUPERLU 222
10.26.2.121 SOLVER_SVFGMRES 223
10.26.2.122 SOLVER_SVGMRES 223
10.26.2.123 SOLVER_UMFPACK 223
10.26.2.124 SOLVER_VBiCGstab 223
10.26.2.125 SOLVER_VFGMRES 223
10.26.2.126 SOLVER_VGMRES 223
10.26.2.127 STAG_RATIO 224
10.26.2.128 STOP_MOD_REL_RES 224
10.26.2.129 STOP_REL_PRECRES 224
10.26.2.130 STOP_REL_RES 224

10.26.2.131 TRUE 224
10.26.2.132 UA_AMG 225
10.26.2.133 UNPT 225
10.26.2.134 USERDEFINED 225
10.26.2.135 V_CYCLE 225
10.26.2.136 VMB 225
10.26.2.137 W_CYCLE 226
10.27 fmgcycle.c File Reference 226
10.27.1 Detailed Description 226
10.27.2 Function Documentation 226
10.27.2.1 fasp_solver_fmgcycle() 226
10.28 formats.c File Reference 227
10.28.1 Detailed Description 228
10.28.2 Function Documentation 228
10.28.2.1 fasp_format_dblc_dcsr() 228
10.28.2.2 fasp_format_dbsr_dcoo() 228
10.28.2.3 fasp_format_dbsr_dcsr() 229
10.28.2.4 fasp_format_dcoo_dcsr() 229
10.28.2.5 fasp_format_dcsr_dbsr() 230
10.28.2.6 fasp_format_dcsr_dcoo() 231
10.28.2.7 fasp_format_dcsrl_dcsr() 231
10.28.2.8 fasp_format_dstr_dbsr() 232
10.28.2.9 fasp_format_dstr_dcsr() 232
10.29 givens.c File Reference 233
10.29.1 Detailed Description 233
10.29.2 Function Documentation 233
10.29.2.1 fasp_aux_givens() 233
10.30 gmg_poisson.c File Reference 234

10.30.1 Detailed Description . . . 235
10.30.2 Function Documentation . . . 235
10.30.2.1 fasp_poisson_fgmg_1D() . . . 235
10.30.2.2 fasp_poisson_fgmg_2D() . . . 236
10.30.2.3 fasp_poisson_fgmg_3D() . . . 236
10.30.2.4 fasp_poisson_gmg_1D() . . . 238
10.30.2.5 fasp_poisson_gmg_2D() . . . 239
10.30.2.6 fasp_poisson_gmg_3D() . . . 240
10.30.2.7 fasp_poisson_pcg_gmg_1D() . . . 240
10.30.2.8 fasp_poisson_pcg_gmg_2D() . . . 241
10.30.2.9 fasp_poisson_pcg_gmg_3D() . . . 242
10.31 graphics.c File Reference . . . 243
10.31.1 Detailed Description . . . 243
10.31.2 Function Documentation . . . 243
10.31.2.1 fasp_dbsr_plot() . . . 243
10.31.2.2 fasp_dbsr_subplot() . . . 244
10.31.2.3 fasp_dcsr_plot() . . . 245
10.31.2.4 fasp_dcsr_subplot() . . . 246
10.31.2.5 fasp_grid2d_plot() . . . 246
10.32 ilu.c File Reference . . . 247
10.32.1 Detailed Description . . . 247
10.32.2 Function Documentation . . . 248
10.32.2.1 fasp_iluk() . . . 248
10.32.2.2 fasp_ilut() . . . 249
10.32.2.3 fasp_ilutp() . . . 250
10.32.2.4 fasp_qsplit() . . . 251
10.32.2.5 fasp_srtr() . . . 251
10.32.2.6 fasp_symbfactor() . . . 252

10.33 ilu_setup_bsr.c File Reference 255
10.33.1 Detailed Description 256
10.33.2 Function Documentation 256
10.33.2.1 fasp_ilu_dbsr_setup() 256
10.33.2.2 fasp_ilu_dbsr_setup_levsch_omp() 256
10.33.2.3 fasp_ilu_dbsr_setup_mc_omp() 257
10.33.2.4 fasp_ilu_dbsr_setup_omp() 258
10.34 ilu_setup_csr.c File Reference 259
10.34.1 Detailed Description 259
10.34.2 Function Documentation 259
10.34.2.1 fasp_ilu_dcsr_setup() 259
10.35 ilu_setup_str.c File Reference 260
10.35.1 Detailed Description 260
10.35.2 Function Documentation 260
10.35.2.1 fasp_ilu_dstr_setup0() 260
10.35.2.2 fasp_ilu_dstr_setup1() 261
10.36 init.c File Reference 262
10.36.1 Detailed Description 262
10.36.2 Function Documentation 262
10.36.2.1 fasp_amg_data_bsr_create() 262
10.36.2.2 fasp_amg_data_bsr_free() 263
10.36.2.3 fasp_amg_data_create() 263
10.36.2.4 fasp_amg_data_free() 264
10.36.2.5 fasp_ilu_data_alloc() 265
10.36.2.6 fasp_ilu_data_free() 265
10.36.2.7 fasp_ilu_data_null() 266
10.36.2.8 fasp_precond_data_null() 266
10.36.2.9 fasp_precond_null() 266

10.36.2.10 fasp_Schwarz_data_free() . . . 267
10.37 input.c File Reference . . . 267
10.37.1 Detailed Description . . . 268
10.37.2 Function Documentation . . . 268
10.37.2.1 fasp_param_check() . . . 268
10.37.2.2 fasp_param_input() . . . 268
10.38 interface_mumps.c File Reference . . . 269
10.38.1 Detailed Description . . . 269
10.38.2 Macro Definition Documentation . . . 270
10.38.2.1 ICNTL . . . 270
10.38.3 Function Documentation . . . 270
10.38.3.1 fasp_solver_mumps() . . . 270
10.38.3.2 fasp_solver_mumps_steps() . . . 271
10.39 interface_pardiso.c File Reference . . . 272
10.39.1 Detailed Description . . . 272
10.39.2 Function Documentation . . . 272
10.39.2.1 fasp_solver_pardiso() . . . 272
10.40 interface_samg.c File Reference . . . 273
10.40.1 Detailed Description . . . 273
10.40.2 Function Documentation . . . 274
10.40.2.1 dCSRmat2SAMGInput() . . . 274
10.40.2.2 dvector2SAMGInput() . . . 274
10.41 interface_superlu.c File Reference . . . 275
10.41.1 Detailed Description . . . 275
10.41.2 Function Documentation . . . 275
10.41.2.1 fasp_solver_superlu() . . . 275
10.42 interface_umfpack.c File Reference . . . 276
10.42.1 Detailed Description . . . 276

10.42.2 Function Documentation 276
10.42.2.1 fasp_solver_umfpack() 276
10.43 interpolation.c File Reference 277
10.43.1 Detailed Description 277
10.43.2 Function Documentation 278
10.43.2.1 fasp_amg_interp() 278
10.43.2.2 fasp_amg_interp1() 278
10.43.2.3 fasp_amg_interp_trunc() 279
10.44 interpolation_em.c File Reference 280
10.44.1 Detailed Description 280
10.44.2 Function Documentation 280
10.44.2.1 fasp_amg_interp_em() 280
10.45 io.c File Reference 281
10.45.1 Detailed Description 283
10.45.2 Function Documentation 283
10.45.2.1 fasp_dbsr_print() 283
10.45.2.2 fasp_dbsr_read() 284
10.45.2.3 fasp_dbsr_write() 285
10.45.2.4 fasp_dbsr_write_coo() 286
10.45.2.5 fasp_dcoo1_read() 287
10.45.2.6 fasp_dcoo_print() 287
10.45.2.7 fasp_dcoo_read() 288
10.45.2.8 fasp_dcoo_shift_read() 288
10.45.2.9 fasp_dcoo_write() 289
10.45.2.10 fasp_dcsr_print() 290
10.45.2.11 fasp_dcsr_read() 290
10.45.2.12 fasp_dcsr_write_coo() 291
10.45.2.13 fasp_dcsrvec1_read() 291

10.45.2.14 fasp_dcsrvec1_write() . . . 292
10.45.2.15 fasp_dcsrvec2_read() . . . 293
10.45.2.16 fasp_dcsrvec2_write() . . . 294
10.45.2.17 fasp_dmtx_read() . . . 295
10.45.2.18 fasp_dmtxsym_read() . . . 295
10.45.2.19 fasp_dstr_print() . . . 296
10.45.2.20 fasp_dstr_read() . . . 297
10.45.2.21 fasp_dstr_write() . . . 297
10.45.2.22 fasp_dvec_print() . . . 298
10.45.2.23 fasp_dvec_read() . . . 298
10.45.2.24 fasp_dvec_write() . . . 299
10.45.2.25 fasp_dvecind_read() . . . 300
10.45.2.26 fasp_dvecind_write() . . . 300
10.45.2.27 fasp_hb_read() . . . 301
10.45.2.28 fasp_ivec_print() . . . 302
10.45.2.29 fasp_ivec_read() . . . 302
10.45.2.30 fasp_ivec_write() . . . 303
10.45.2.31 fasp_ivecind_read() . . . 303
10.45.2.32 fasp_matrix_read() . . . 304
10.45.2.33 fasp_matrix_read_bin() . . . 305
10.45.2.34 fasp_matrix_write() . . . 305
10.45.2.35 fasp_vector_read() . . . 306
10.45.2.36 fasp_vector_write() . . . 307
10.45.3 Variable Documentation . . . 308
10.45.3.1 dlength . . . 308
10.45.3.2 ilength . . . 308
10.46 itsolver_blc.c File Reference . . . 309
10.46.1 Detailed Description . . . 309

10.46.2 Function Documentation 309
10.46.2.1 fasp_solver_dblc_itsolver() 309
10.46.2.2 fasp_solver_dblc_krylov() 310
10.46.2.3 fasp_solver_dblc_krylov_block_3() 311
10.46.2.4 fasp_solver_dblc_krylov_block_4() 312
10.46.2.5 fasp_solver_dblc_krylov_sweeping() 312
10.47itsolver_bsr.c File Reference 313
10.47.1 Detailed Description 314
10.47.2 Function Documentation 314
10.47.2.1 fasp_solver_dbsr_itsolver() 314
10.47.2.2 fasp_solver_dbsr_krylov() 315
10.47.2.3 fasp_solver_dbsr_krylov_amg() 315
10.47.2.4 fasp_solver_dbsr_krylov_amg_nk() 316
10.47.2.5 fasp_solver_dbsr_krylov_diag() 317
10.47.2.6 fasp_solver_dbsr_krylov_ilu() 318
10.47.2.7 fasp_solver_dbsr_krylov_nk_amg() 319
10.48itsolver_csr.c File Reference 320
10.48.1 Detailed Description 321
10.48.2 Function Documentation 321
10.48.2.1 fasp_solver_dcsr_itsolver() 321
10.48.2.2 fasp_solver_dcsr_krylov() 322
10.48.2.3 fasp_solver_dcsr_krylov_amg() 322
10.48.2.4 fasp_solver_dcsr_krylov_amg_nk() 323
10.48.2.5 fasp_solver_dcsr_krylov_diag() 324
10.48.2.6 fasp_solver_dcsr_krylov_ilu() 324
10.48.2.7 fasp_solver_dcsr_krylov_ilu_M() 325
10.48.2.8 fasp_solver_dcsr_krylov_Schwarz() 326
10.49itsolver_mf.c File Reference 327

10.49.1 Detailed Description 327
10.49.2 Function Documentation 327
10.49.2.1 fasp_solver_itsolver() 327
10.49.2.2 fasp_solver_itsolver_init() 328
10.49.2.3 fasp_solver_krylov() 329
10.50 itsolver_str.c File Reference 330
10.50.1 Detailed Description 330
10.50.2 Function Documentation 330
10.50.2.1 fasp_solver_dstr_itsolver() 330
10.50.2.2 fasp_solver_dstr_krylov() 331
10.50.2.3 fasp_solver_dstr_krylov_blockgs() 332
10.50.2.4 fasp_solver_dstr_krylov_diag() 332
10.50.2.5 fasp_solver_dstr_krylov_ilu() 333
10.51 lu.c File Reference 334
10.51.1 Detailed Description 334
10.51.2 Function Documentation 334
10.51.2.1 fasp_smat_lu_decomp() 334
10.51.2.2 fasp_smat_lu_solve() 335
10.52 memory.c File Reference 336
10.52.1 Detailed Description 337
10.52.2 Function Documentation 337
10.52.2.1 fasp_mem_calloc() 337
10.52.2.2 fasp_mem_check() 337
10.52.2.3 fasp_mem_dcsr_check() 338
10.52.2.4 fasp_mem_free() 339
10.52.2.5 fasp_mem_iludata_check() 339
10.52.2.6 fasp_mem_realloc() 340
10.52.2.7 fasp_mem_usage() 340

10.52.3 Variable Documentation . . . 341
10.52.3.1 total_alloc_count . . . 341
10.52.3.2 total_alloc_mem . . . 341
10.53 message.c File Reference . . . 341
10.53.1 Detailed Description . . . 342
10.53.2 Function Documentation . . . 342
10.53.2.1 fasp_chkerr() . . . 342
10.53.2.2 print_amgcomplexity() . . . 342
10.53.2.3 print_amgcomplexity_bsr() . . . 343
10.53.2.4 print_cputime() . . . 343
10.53.2.5 print_itinfo() . . . 344
10.53.2.6 print_message() . . . 344
10.54 mgcycle.c File Reference . . . 345
10.54.1 Detailed Description . . . 345
10.54.2 Function Documentation . . . 345
10.54.2.1 fasp_solver_mgcycle() . . . 345
10.54.2.2 fasp_solver_mgcycle_bsr() . . . 346
10.55 mgrecur.c File Reference . . . 346
10.55.1 Detailed Description . . . 347
10.55.2 Function Documentation . . . 347
10.55.2.1 fasp_solver_mgrecur() . . . 347
10.56 ordering.c File Reference . . . 348
10.56.1 Detailed Description . . . 348
10.56.2 Function Documentation . . . 348
10.56.2.1 fasp_aux_dQuickSort() . . . 348
10.56.2.2 fasp_aux_dQuickSortIndex() . . . 349
10.56.2.3 fasp_aux_iQuickSort() . . . 350
10.56.2.4 fasp_aux_iQuickSortIndex() . . . 350

10.56.2.5 fasp_aux_merge() . . . 351
10.56.2.6 fasp_aux_msort() . . . 352
10.56.2.7 fasp_aux_unique() . . . 352
10.56.2.8 fasp_BinarySearch() . . . 353
10.56.2.9 fasp_dcsr_CMK_order() . . . 354
10.56.2.10 fasp_dcsr_RCMK_order() . . . 354
10.56.2.11 fasp_multicolors_independent_set() . . . 355
10.56.2.12 fasp_topological_sorting_ilu() . . . 355
10.57 parameters.c File Reference . . . 356
10.57.1 Detailed Description . . . 357
10.57.2 Function Documentation . . . 357
10.57.2.1 fasp_param_amg_init() . . . 357
10.57.2.2 fasp_param_amg_print() . . . 358
10.57.2.3 fasp_param_amg_set() . . . 359
10.57.2.4 fasp_param_amg_to_prec() . . . 359
10.57.2.5 fasp_param_amg_to_prec_bsr() . . . 360
10.57.2.6 fasp_param_ilu_init() . . . 360
10.57.2.7 fasp_param_ilu_print() . . . 361
10.57.2.8 fasp_param_ilu_set() . . . 361
10.57.2.9 fasp_param_init() . . . 362
10.57.2.10 fasp_param_input_init() . . . 363
10.57.2.11 fasp_param_prec_to_amg() . . . 363
10.57.2.12 fasp_param_prec_to_amg_bsr() . . . 364
10.57.2.13 fasp_param_Schwarz_init() . . . 364
10.57.2.14 fasp_param_Schwarz_print() . . . 365
10.57.2.15 fasp_param_Schwarz_set() . . . 365
10.57.2.16 fasp_param_set() . . . 366
10.57.2.17 fasp_param_solver_init() . . . 366

10.57.2.18 fasp_param_solver_print() 367
10.57.2.19 fasp_param_solver_set() 367
10.58 pbcgs.c File Reference 368
10.58.1 Detailed Description 368
10.58.2 Function Documentation 370
10.58.2.1 fasp_solver_dblc_pbcgs() 370
10.58.2.2 fasp_solver_dblc_pvbcgs() 371
10.58.2.3 fasp_solver_dbsr_pbcgs() 371
10.58.2.4 fasp_solver_dbsr_pvbcgs() 372
10.58.2.5 fasp_solver_dcsr_pbcgs() 373
10.58.2.6 fasp_solver_dcsr_pvbcgs() 374
10.58.2.7 fasp_solver_dstr_pbcgs() 375
10.58.2.8 fasp_solver_dstr_pvbcgs() 376
10.59 pbcgs_mf.c File Reference 376
10.59.1 Detailed Description 377
10.59.2 Function Documentation 378
10.59.2.1 fasp_solver_pbcgs() 378
10.59.2.2 fasp_solver_pvbcgs() 379
10.60 pcg.c File Reference 380
10.60.1 Detailed Description 380
10.60.2 Function Documentation 381
10.60.2.1 fasp_solver_dblc_pcg() 381
10.60.2.2 fasp_solver_dbsr_pcg() 382
10.60.2.3 fasp_solver_dcsr_pcg() 383
10.60.2.4 fasp_solver_dstr_pcg() 384
10.61 pcg_mf.c File Reference 385
10.61.1 Detailed Description 385
10.61.2 Function Documentation 386

10.61.2.1 fasp_solver_pcg() 386
10.62pgcg.c File Reference 387
10.62.1 Detailed Description 387
10.62.2 Function Documentation 388
10.62.2.1 fasp_solver_dcsr_pgcg() 388
10.63pgcg_mf.c File Reference 388
10.63.1 Detailed Description 389
10.63.2 Function Documentation 389
10.63.2.1 fasp_solver_pgcg() 389
10.64pgcr.c File Reference 390
10.64.1 Detailed Description 390
10.64.2 Function Documentation 391
10.64.2.1 fasp_solver_dcsr_pgcr() 391
10.64.2.2 fasp_solver_dcsr_pgcr1() 392
10.65pgmres.c File Reference 393
10.65.1 Detailed Description 394
10.65.2 Function Documentation 394
10.65.2.1 fasp_solver_dblc_pgmres() 394
10.65.2.2 fasp_solver_dbsr_pgmres() 395
10.65.2.3 fasp_solver_dcsr_pgmres() 396
10.65.2.4 fasp_solver_dstr_pgmres() 397
10.66pgmres_mf.c File Reference 398
10.66.1 Detailed Description 398
10.66.2 Function Documentation 398
10.66.2.1 fasp_solver_pgmres() 398
10.67pminres.c File Reference 399
10.67.1 Detailed Description 400
10.67.2 Function Documentation 401

10.67.2.1 fasp_solver_dblc_pminres() 401
10.67.2.2 fasp_solver_dcsr_pminres() 402
10.67.2.3 fasp_solver_dstr_pminres() 403
10.68 pminres_mf.c File Reference 403
10.68.1 Detailed Description 404
10.68.2 Function Documentation 405
10.68.2.1 fasp_solver_pminres() 405
10.69 precond_blc.c File Reference 406
10.69.1 Detailed Description 407
10.69.2 Function Documentation 407
10.69.2.1 fasp_precond_block_diag_3() 407
10.69.2.2 fasp_precond_block_diag_3_amg() 407
10.69.2.3 fasp_precond_block_diag_4() 408
10.69.2.4 fasp_precond_block_lower_3() 408
10.69.2.5 fasp_precond_block_lower_3_amg() 410
10.69.2.6 fasp_precond_block_lower_4() 410
10.69.2.7 fasp_precond_block_SGS_3() 412
10.69.2.8 fasp_precond_block_SGS_3_amg() 412
10.69.2.9 fasp_precond_block_upper_3() 414
10.69.2.10 fasp_precond_block_upper_3_amg() 414
10.69.2.11 fasp_precond_sweeping() 416
10.70 precond_bsr.c File Reference 416
10.70.1 Detailed Description 417
10.70.2 Function Documentation 417
10.70.2.1 fasp_precond_dbsr_amg() 417
10.70.2.2 fasp_precond_dbsr_amg_nk() 418
10.70.2.3 fasp_precond_dbsr_diag() 418
10.70.2.4 fasp_precond_dbsr_diag_nc2() 420

10.70.2.5 fasp_precond_dbsr_diag_nc3() 421
10.70.2.6 fasp_precond_dbsr_diag_nc5() 422
10.70.2.7 fasp_precond_dbsr_diag_nc7() 422
10.70.2.8 fasp_precond_dbsr_ilu() 423
10.70.2.9 fasp_precond_dbsr_ilu_levsch_omp() 424
10.70.2.10 fasp_precond_dbsr_ilu_mc_omp() 424
10.70.2.11 fasp_precond_dbsr_nl_amli() 425
10.71 precond_csr.c File Reference 426
10.71.1 Detailed Description 426
10.71.2 Function Documentation 426
10.71.2.1 fasp_precond_amg() 426
10.71.2.2 fasp_precond_amg_nk() 427
10.71.2.3 fasp_precond_amli() 427
10.71.2.4 fasp_precond_diag() 429
10.71.2.5 fasp_precond_famg() 429
10.71.2.6 fasp_precond_free() 431
10.71.2.7 fasp_precond_ilu() 432
10.71.2.8 fasp_precond_ilu_backward() 432
10.71.2.9 fasp_precond_ilu_forward() 433
10.71.2.10 fasp_precond_nl_amli() 433
10.71.2.11 fasp_precond_Schwarz() 434
10.71.2.12 fasp_precond_setup() 434
10.72 precond_str.c File Reference 435
10.72.1 Detailed Description 436
10.72.2 Function Documentation 436
10.72.2.1 fasp_precond_dstr_blockgs() 436
10.72.2.2 fasp_precond_dstr_diag() 436
10.72.2.3 fasp_precond_dstr_ilu0() 437

10.72.2.4 fasp_precond_dstr_ilu0_backward() 437
10.72.2.5 fasp_precond_dstr_ilu0_forward() 438
10.72.2.6 fasp_precond_dstr_ilu1() 439
10.72.2.7 fasp_precond_dstr_ilu1_backward() 440
10.72.2.8 fasp_precond_dstr_ilu1_forward() 441
10.73pvfgmres.c File Reference 442
10.73.1 Detailed Description 443
10.73.2 Function Documentation 443
10.73.2.1 fasp_solver_dblc_pvfgmres() 443
10.73.2.2 fasp_solver_dbsr_pvfgmres() 444
10.73.2.3 fasp_solver_dcsr_pvfgmres() 445
10.74pvfgmres_mf.c File Reference 446
10.74.1 Detailed Description 446
10.74.2 Function Documentation 446
10.74.2.1 fasp_solver_pvfgmres() 446
10.75pvgmres.c File Reference 447
10.75.1 Detailed Description 448
10.75.2 Function Documentation 448
10.75.2.1 fasp_solver_dblc_pvgmres() 448
10.75.2.2 fasp_solver_dbsr_pvgmres() 449
10.75.2.3 fasp_solver_dcsr_pvgmres() 450
10.75.2.4 fasp_solver_dstr_pvgmres() 451
10.76pvgmres_mf.c File Reference 451
10.76.1 Detailed Description 452
10.76.2 Function Documentation 452
10.76.2.1 fasp_solver_pvgmres() 452
10.77quadrature.c File Reference 453
10.77.1 Detailed Description 453

10.77.2 Function Documentation 453
10.77.2.1 fasp_gauss2d() 453
10.77.2.2 fasp_quad2d() 454
10.78rap.c File Reference 455
10.78.1 Detailed Description 455
10.78.2 Function Documentation 455
10.78.2.1 fasp_blas_dcsr_rap2() 455
10.79schwarz_setup.c File Reference 456
10.79.1 Detailed Description 456
10.79.2 Function Documentation 456
10.79.2.1 fasp_dcsr_Schwarz_backward_smoother() 456
10.79.2.2 fasp_dcsr_Schwarz_forward_smoother() 457
10.79.2.3 fasp_Schwarz_get_block_matrix() 458
10.79.2.4 fasp_Schwarz_setup() 458
10.80smat.c File Reference 459
10.80.1 Detailed Description 460
10.80.2 Macro Definition Documentation 460
10.80.2.1 SWAP 460
10.80.3 Function Documentation 460
10.80.3.1 fasp_blas_smat_inv() 460
10.80.3.2 fasp_blas_smat_inv_nc() 461
10.80.3.3 fasp_blas_smat_inv_nc2() 461
10.80.3.4 fasp_blas_smat_inv_nc3() 462
10.80.3.5 fasp_blas_smat_inv_nc4() 462
10.80.3.6 fasp_blas_smat_inv_nc5() 463
10.80.3.7 fasp_blas_smat_inv_nc7() 463
10.80.3.8 fasp_blas_smat_invp_nc() 464
10.80.3.9 fasp_blas_smat_Linfinity() 464

10.80.3.10 fasp_iden_free() 465
10.80.3.11 fasp_smat_identity() 465
10.80.3.12 fasp_smat_identity_nc2() 466
10.80.3.13 fasp_smat_identity_nc3() 466
10.80.3.14 fasp_smat_identity_nc5() 467
10.80.3.15 fasp_smat_identity_nc7() 467
10.81 smoother_bsr.c File Reference 468
10.81.1 Detailed Description 469
10.81.2 Function Documentation 469
10.81.2.1 fasp_smoother_dbsr_gs() 469
10.81.2.2 fasp_smoother_dbsr_gs1() 470
10.81.2.3 fasp_smoother_dbsr_gs_ascend() 471
10.81.2.4 fasp_smoother_dbsr_gs_ascend1() 471
10.81.2.5 fasp_smoother_dbsr_gs_descend() 472
10.81.2.6 fasp_smoother_dbsr_gs_descend1() 472
10.81.2.7 fasp_smoother_dbsr_gs_order1() 473
10.81.2.8 fasp_smoother_dbsr_gs_order2() 474
10.81.2.9 fasp_smoother_dbsr_ilu() 474
10.81.2.10 fasp_smoother_dbsr_jacobi() 475
10.81.2.11 fasp_smoother_dbsr_jacobi1() 476
10.81.2.12 fasp_smoother_dbsr_jacobi_setup() 476
10.81.2.13 fasp_smoother_dbsr_sor() 477
10.81.2.14 fasp_smoother_dbsr_sor1() 478
10.81.2.15 fasp_smoother_dbsr_sor_ascend() 478
10.81.2.16 fasp_smoother_dbsr_sor_descend() 479
10.81.2.17 fasp_smoother_dbsr_sor_order() 480
10.81.3 Variable Documentation 480
10.81.3.1 ilu_solve_omp 480

10.82 smoother_csr.c File Reference . . . 481
10.82.1 Detailed Description . . . 481
10.82.2 Function Documentation . . . 482
10.82.2.1 fasp_smoother_dcsr_gs() . . . 482
10.82.2.2 fasp_smoother_dcsr_gs_cf() . . . 482
10.82.2.3 fasp_smoother_dcsr_gs_rb3d() . . . 483
10.82.2.4 fasp_smoother_dcsr_ilu() . . . 484
10.82.2.5 fasp_smoother_dcsr_jacobi() . . . 484
10.82.2.6 fasp_smoother_dcsr_kaczmarz() . . . 485
10.82.2.7 fasp_smoother_dcsr_L1diag() . . . 486
10.82.2.8 fasp_smoother_dcsr_sgs() . . . 487
10.82.2.9 fasp_smoother_dcsr_sor() . . . 487
10.82.2.10 fasp_smoother_dcsr_sor_cf() . . . 488
10.83 smoother_csr_cr.c File Reference . . . 489
10.83.1 Detailed Description . . . 489
10.83.2 Function Documentation . . . 489
10.83.2.1 fasp_smoother_dcsr_gscr() . . . 489
10.84 smoother_csr_poly.c File Reference . . . 490
10.84.1 Detailed Description . . . 491
10.84.2 Function Documentation . . . 491
10.84.2.1 fasp_smoother_dcsr_poly() . . . 491
10.84.2.2 fasp_smoother_dcsr_poly_old() . . . 491
10.85 smoother_str.c File Reference . . . 493
10.85.1 Detailed Description . . . 494
10.85.2 Function Documentation . . . 494
10.85.2.1 fasp_generate_diaginv_block() . . . 494
10.85.2.2 fasp_smoother_dstr_gs() . . . 495
10.85.2.3 fasp_smoother_dstr_gs1() . . . 495

10.85.2.4 fasp_smoother_dstr_gs_ascend() 496
10.85.2.5 fasp_smoother_dstr_gs_cf() 497
10.85.2.6 fasp_smoother_dstr_gs_descend() 497
10.85.2.7 fasp_smoother_dstr_gs_order() 499
10.85.2.8 fasp_smoother_dstr_jacobi() 500
10.85.2.9 fasp_smoother_dstr_jacobi1() 500
10.85.2.10 fasp_smoother_dstr_schwarz() 501
10.85.2.11 fasp_smoother_dstr_sor() 501
10.85.2.12 fasp_smoother_dstr_sor1() 502
10.85.2.13 fasp_smoother_dstr_sor_ascend() 503
10.85.2.14 fasp_smoother_dstr_sor_cf() 503
10.85.2.15 fasp_smoother_dstr_sor_descend() 505
10.85.2.16 fasp_smoother_dstr_sor_order() 506
10.86 sparse_block.c File Reference 506
10.86.1 Detailed Description 507
10.86.2 Function Documentation 507
10.86.2.1 fasp_dblc_free() 507
10.86.2.2 fasp_dbsr_getblk() 507
10.86.2.3 fasp_dbsr_getblk_dcsr() 508
10.86.2.4 fasp_dbsr_Linfinity_dcsr() 509
10.86.2.5 fasp_dcsr_getblk() 509
10.87 sparse_bsr.c File Reference 510
10.87.1 Detailed Description 511
10.87.2 Function Documentation 511
10.87.2.1 fasp_dbsr_alloc() 511
10.87.2.2 fasp_dbsr_cp() 512
10.87.2.3 fasp_dbsr_create() 513
10.87.2.4 fasp_dbsr_diaginv() 513

10.87.2.5 fasp_dbsr_diaginv2() 514
10.87.2.6 fasp_dbsr_diaginv3() 515
10.87.2.7 fasp_dbsr_diaginv4() 516
10.87.2.8 fasp_dbsr_diagLU() 517
10.87.2.9 fasp_dbsr_diagLU2() 518
10.87.2.10 fasp_dbsr_diagpref() 518
10.87.2.11 fasp_dbsr_free() 519
10.87.2.12 fasp_dbsr_getdiag() 519
10.87.2.13 fasp_dbsr_getdiaginv() 520
10.87.2.14 fasp_dbsr_null() 521
10.87.2.15 fasp_dbsr_perm() 521
10.87.2.16 fasp_dbsr_trans() 522
10.88 sparse_coo.c File Reference 522
10.88.1 Detailed Description 523
10.88.2 Function Documentation 523
10.88.2.1 fasp_dcoo_alloc() 523
10.88.2.2 fasp_dcoo_create() 524
10.88.2.3 fasp_dcoo_free() 524
10.88.2.4 fasp_dcoo_shift() 525
10.89 sparse_csr.c File Reference 525
10.89.1 Detailed Description 527
10.89.2 Function Documentation 527
10.89.2.1 fasp_dcsr_alloc() 527
10.89.2.2 fasp_dcsr_compress() 528
10.89.2.3 fasp_dcsr_compress_inplace() 528
10.89.2.4 fasp_dcsr_cp() 529
10.89.2.5 fasp_dcsr_create() 529
10.89.2.6 fasp_dcsr_diagpref() 530

10.89.2.7 fasp_dcsr_free() . . . 531
10.89.2.8 fasp_dcsr_getcol() . . . 531
10.89.2.9 fasp_dcsr_getdiag() . . . 532
10.89.2.10 fasp_dcsr_multicoloring() . . . 532
10.89.2.11 fasp_dcsr_null() . . . 533
10.89.2.12 fasp_dcsr_perm() . . . 533
10.89.2.13 fasp_dcsr_permz() . . . 534
10.89.2.14 fasp_dcsr_regdiag() . . . 535
10.89.2.15 fasp_dcsr_shift() . . . 535
10.89.2.16 fasp_dcsr_sort() . . . 536
10.89.2.17 fasp_dcsr_sortz() . . . 536
10.89.2.18 fasp_dcsr_symdiagscale() . . . 537
10.89.2.19 fasp_dcsr_sympat() . . . 538
10.89.2.20 fasp_dcsr_trans() . . . 538
10.89.2.21 fasp_dcsr_transz() . . . 539
10.89.2.22 fasp_icsr_cp() . . . 539
10.89.2.23 fasp_icsr_create() . . . 540
10.89.2.24 fasp_icsr_free() . . . 541
10.89.2.25 fasp_icsr_null() . . . 542
10.89.2.26 fasp_icsr_trans() . . . 542
10.90 sparse_csrl.c File Reference . . . 543
10.90.1 Detailed Description . . . 543
10.90.2 Function Documentation . . . 544
10.90.2.1 fasp_dcsrl_create() . . . 544
10.90.2.2 fasp_dcsrl_free() . . . 545
10.91 sparse_str.c File Reference . . . 545
10.91.1 Detailed Description . . . 546
10.91.2 Function Documentation . . . 546

10.91.2.1 fasp_dstr_alloc() 546
10.91.2.2 fasp_dstr_cp() 547
10.91.2.3 fasp_dstr_create() 547
10.91.2.4 fasp_dstr_free() 548
10.91.2.5 fasp_dstr_null() 549
10.92 sparse_util.c File Reference 549
10.92.1 Detailed Description 550
10.92.2 Function Documentation 550
10.92.2.1 fasp_sparse_aat_() 550
10.92.2.2 fasp_sparse_abyb_() 551
10.92.2.3 fasp_sparse_abybms_() 552
10.92.2.4 fasp_sparse_aplbms_() 553
10.92.2.5 fasp_sparse_aplusb_() 553
10.92.2.6 fasp_sparse_iit_() 554
10.92.2.7 fasp_sparse_MIS() 554
10.92.2.8 fasp_sparse_rapcmp_() 555
10.92.2.9 fasp_sparse_rapms_() 556
10.92.2.10 fasp_sparse_wta_() 557
10.92.2.11 fasp_sparse_wtams_() 557
10.92.2.12 fasp_sparse_ytx_() 558
10.92.2.13 fasp_sparse_ytxbig_() 559
10.93 spbcgs.c File Reference 559
10.93.1 Detailed Description 560
10.93.2 Function Documentation 561
10.93.2.1 fasp_solver_dblc_spbcgs() 561
10.93.2.2 fasp_solver_dbsr_spbcgs() 562
10.93.2.3 fasp_solver_dcsr_spbcgs() 563
10.93.2.4 fasp_solver_dstr_spbcgs() 564

10.94 spcg.c File Reference . . . 565
10.94.1 Detailed Description . . . 565
10.94.2 Function Documentation . . . 566
10.94.2.1 fasp_solver_dblc_spcg() . . . 566
10.94.2.2 fasp_solver_dcsr_spcg() . . . 567
10.94.2.3 fasp_solver_dstr_spcg() . . . 568
10.95 spgmres.c File Reference . . . 569
10.95.1 Detailed Description . . . 569
10.95.2 Function Documentation . . . 570
10.95.2.1 fasp_solver_dblc_spgmres() . . . 570
10.95.2.2 fasp_solver_dbsr_spgmres() . . . 571
10.95.2.3 fasp_solver_dcsr_spgmres() . . . 571
10.95.2.4 fasp_solver_dstr_spgmres() . . . 572
10.96 spminres.c File Reference . . . 573
10.96.1 Detailed Description . . . 574
10.96.2 Function Documentation . . . 575
10.96.2.1 fasp_solver_dblc_spminres() . . . 575
10.96.2.2 fasp_solver_dcsr_spminres() . . . 576
10.96.2.3 fasp_solver_dstr_spminres() . . . 577
10.97 spvgmres.c File Reference . . . 578
10.97.1 Detailed Description . . . 578
10.97.2 Function Documentation . . . 578
10.97.2.1 fasp_solver_dblc_spvgmres() . . . 578
10.97.2.2 fasp_solver_dbsr_spvgmres() . . . 580
10.97.2.3 fasp_solver_dcsr_spvgmres() . . . 581
10.97.2.4 fasp_solver_dstr_spvgmres() . . . 582
10.98 threads.c File Reference . . . 583
10.98.1 Detailed Description . . . 583

10.98.2 Function Documentation . . . 583
10.98.2.1 FASP_GET_START_END() . . . 583
10.98.2.2 fasp_set_GS_threads() . . . 584
10.98.3 Variable Documentation . . . 584
10.98.3.1 THDs_AMG_GS . . . 584
10.98.3.2 THDs_CPR_gGS . . . 585
10.98.3.3 THDs_CPR_lGS . . . 585
10.99 timing.c File Reference . . . 585
10.99.1 Detailed Description . . . 585
10.99.2 Function Documentation . . . 585
10.99.2.1 fasp_gettime() . . . 585
10.100 vec.c File Reference . . . 586
10.100.1 Detailed Description . . . 586
10.100.2 Function Documentation . . . 587
10.100.2.1 fasp_dvec_alloc() . . . 587
10.100.2.2 fasp_dvec_cp() . . . 587
10.100.2.3 fasp_dvec_create() . . . 588
10.100.2.4 fasp_dvec_free() . . . 588
10.100.2.5 fasp_dvec_isnan() . . . 589
10.100.2.6 fasp_dvec_maxdiff() . . . 589
10.100.2.7 fasp_dvec_null() . . . 590
10.100.2.8 fasp_dvec_rand() . . . 590
10.100.2.9 fasp_dvec_set() . . . 591
10.100.2.10 fasp_dvec_symdiagscale() . . . 592
10.100.2.11 fasp_ivec_alloc() . . . 592
10.100.2.12 fasp_ivec_create() . . . 593
10.100.2.13 fasp_ivec_free() . . . 593
10.100.2.14 fasp_ivec_set() . . . 594
10.101 wrapper.c File Reference . . . 595
10.101.1 Detailed Description . . . 595
10.101.2 Function Documentation . . . 595
10.101.2.1 fasp_fwrapper_amg_() . . . 595
10.101.2.2 fasp_fwrapper_krylov_amg_() . . . 596
10.101.2.3 fasp_wrapper_dbsr_krylov_amg() . . . 597
10.101.2.4 fasp_wrapper_dcoo_dbsr_krylov_amg() . . . 598

Index **599**

Chapter 1

Introduction

Over the last few decades, researchers have expended significant effort on developing efficient iterative methods for solving discretized partial differential equations (PDEs). Though these efforts have yielded many mathematically optimal solvers such as the multigrid method, the unfortunate reality is that multigrid methods have not been much used in practical applications. This marked gap between theory and practice is mainly due to the fragility of traditional multigrid (MG) methodology and the complexity of its implementation. We aim to develop techniques and the corresponding software that will narrow this gap, specifically by developing mathematically optimal solvers that are robust and easy to use in practice.

We believe that there is no one-size-for-all solution method for discrete linear systemsfrom different applications. And, efficient iterative solvers can be constructed by taking the properties of PDEs and discretizations into account. In this project, we plan to construct a pool of discrete problems arising from partial differential equations (PDEs) or PDE systems and efficient linear solvers for these problems. We mainly utilize the methodology of Auxiliary Space Preconditioning (ASP) to construct efficient linear solvers. Due to this reason, this software package is called Fast Auxiliary Space Preconditioning or FASP for short.

FASP contains the kernel part and several applications (ranging from fluid dynamics to reservoir simulation). The kernel part is open-source and licensed under GNU Lesser General Public License or LGPL version 3.0 or later. Some of the applications contain contributions from and owned partially by other parties.

> For the moment, FASP is under alpha testing. If you wish to obtain a current version of FASP or you have any questions, feel free to contact us at `faspdev@gmail.com`.

This software distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

Chapter 2

How to obtain FASP

The most updated version of FASP can be downloaded from

http://fasp.sourceforge.net/download/faspsolver.zip

We use HG (Mecurial) as our main version control tool. HG is easy to use and it is available at all OS platforms. For people who is interested in the developer version, you can obtain the FASP package with hg:

$ hg clone https://faspusers@bitbucket.org/fasp/faspsolver

will give you the developer version of the FASP package.

Chapter 3

Building and Installation

This is a simple instruction on building and testing. For more details, please refer to the README files and the short `User's Guide` in "faspsolver/doc/".

To compile, you need a Fortran and a C compiler. First, you can type in the "faspsolver/" root directory:

```
$ make config
```

which will config the environment automatically. And, then, you can need to type:

```
$ make install
```

which will make the FASP shared static library and install to PREFIX/. By default, FASP libraries and executables will be installed in the FASP home directory "faspsolver/".

There is a simple GUI tool for building and installing FASP included in the package. You need Tcl/Tk support in your computer. You may call this GUI by run in the root directory:

```
$ wish fasp_install.tcl
```

If you need to see the detailed usage of "make" or need any help, please type:

```
$ make help
```

After installation, tutorial examples can be found in "tutorial/".

Chapter 4

Developers

Project leader:

- Xu, Jinchao (Penn State University, USA)

Project coordinator:

- Zhang, Chensong (Chinese Academy of Sciences, China)

Current active developers (in alphabetic order):

- Feng, Chunsheng (Xiangtan University, China)
- Hu, Xiaozhe (Tufts University, USA)
- Li, Zheng (Kunming University of Science and Technology, China)
- Zhang, Chensong (Chinese Academy of Sciences, China)
- Zhang, Hongxuan (Penn State Univeristy, USA)

With contributions from (in alphabetic order):

- Brannick, James (Penn State University, USA)
- Chen, Long (University of California, Irvine, USA)
- Huang, Feiteng (Sichuang University, China)
- Huang, Xuehai (Shanghai Jiaotong University, China)
- Qiao, Changhe (Penn State University, USA)
- Shu, Shi (Xiangtan University, China)
- Sun, Pengtao (University of Nevada, Las Vegas, USA)

- Yang, Kai (Penn State University, USA)
- Yue, Xiaoqiang (Xiangtan University, China)
- Wang, Lu (LLNL, USA)
- Wang, Ziteng (University of Alabama, USA)
- Zhang, Shiquan (Sichuan University, China)
- Zhang, Shuo (Chinese Academy of Sciences, China)
- Zhang, Weifeng (Kunming University of Science and Technology, China)
- Zhou, Zhiyang (Xiangtan University, China)

Chapter 5

Doxygen

We use Doxygen as our automatically documentation generator which will make our future maintainance minimized. You can obtain the software (Windows, Linux and OS X) as well as its manual on the official website

`http://www.doxygen.org`

For an ordinary user, Doxygen is completely trivial to use. We only need to use some special marker in the usual comment as we put in c-files.

Chapter 6

Todo List

File sparse_util.c

Remove unwanted functions from this file. –Chensong

Chapter 7

Data Structure Index

7.1 Data Structures

Here are the data structures with brief descriptions:

AMG_data
Data for AMG solvers . . . 23
AMG_data_bsr
Data for multigrid levels. (BSR format) . . . 24
AMG_param
Parameters for AMG solver . . . 26
block_BSR
Block REAL matrix format for reservoir simulation . . . 28
block_dvector
Block REAL vector structure . . . 29
block_iCSRmat
Block INT CSR matrix format . . . 30
block_ivector
Block INT vector structure . . . 30
block_Reservoir
Block REAL matrix format for reservoir simulation . . . 31
dBLCmat
Block REAL CSR matrix format . . . 32
dBSRmat
Block sparse row storage matrix of REAL type . . . 32
dCOOmat
Sparse matrix of REAL type in COO (or IJ) format . . . 33
dCSRLmat
Sparse matrix of REAL type in CSRL format . . . 34
dCSRmat
Sparse matrix of REAL type in CSR format . . . 35
ddenmat
Dense matrix of REAL type . . . 36
dSTRmat
Structure matrix of REAL type . . . 37
dvector
Vector with n entries of REAL type . . . 38

grid2d
Two dimensional grid data structure 38
iCOOmat
Sparse matrix of INT type in COO (or IJ) format 41
iCSRmat
Sparse matrix of INT type in CSR format 42
idenmat
Dense matrix of INT type 43
ILU_data
Data for ILU setup 43
ILU_param
Parameters for ILU 45
input_param
Input parameters 45
itsolver_param
Parameters passed to iterative solvers 56
ivector
Vector with n entries of INT type 58
Link
Struct for Links 58
linked_list
A linked list node 59
mallinfo 60
malloc_chunk 60
malloc_params 61
malloc_segment 61
malloc_state 62
malloc_tree_chunk 62
Mumps_data
Parameters for MUMPS interface 63
mxv_matfree
Matrix-vector multiplication, replace the actual matrix 63
nedmallinfo 64
Pardiso_data
Parameters for Intel MKL PARDISO interface 64
precond
Preconditioner data and action 65
precond_block_data
Data passed to the preconditioner for block preconditioning for dBLCmat format 65
precond_block_reservoir_data
Data passed to the preconditioner for reservoir simulation problems 67
precond_data
Data passed to the preconditioners 72
precond_data_bsr
Data passed to the preconditioners 73
precond_data_str
Data passed to the preconditioner for dSTRmat matrices 75
precond_diagbsr
Data passed to diagnal preconditioner for dBSRmat matrices 77
precond_diagstr
Data passed to diagonal preconditioner for dSTRmat matrices 77
precond_FASP_blkoil_data
Data passed to the preconditioner for preconditioning reservoir simulation problems 78

precond_sweeping_data
Data passed to the preconditioner for sweeping preconditioning 84
Schwarz_data
Data for Schwarz methods 86
Schwarz_param
Parameters for Schwarz method 88

Chapter 8

File Index

8.1 File List

Here is a list of all documented files with brief descriptions:

amg.c
AMG method as an iterative solver (main file) . . . 89
amg_setup_cr.c
Brannick-Falgout compatible relaxation based AMG: SETUP phase . . . 90
amg_setup_rs.c
Ruge-Stuben AMG: SETUP phase . . . 92
amg_setup_sa.c
Smoothed aggregation AMG: SETUP phase . . . 93
amg_setup_ua.c
Unsmoothed aggregation AMG: SETUP phase . . . 95
amg_solve.c
Algebraic multigrid iterations: SOLVE phase . . . 96
amlirecur.c
Abstract AMLI multilevel iteration – recursive version . . . 100
array.c
Simple array operations – init, set, copy, etc . . . 103
blas_array.c
BLAS1 operations for arrays . . . 109
blas_blc.c
BLAS2 operations for dBLCmat matrices . . . 115
blas_bsr.c
BLAS2 operations for dBSRmat matrices . . . 118
blas_csr.c
BLAS2 operations for dCSRmat matrices . . . 125
blas_csrl.c
BLAS2 operations for dCSRLmat matrices . . . 135
blas_smat.c
BLAS2 operations for *small* dense matrices . . . 136
blas_str.c
BLAS2 operations for dSTRmat matrices . . . 162
blas_vec.c
BLAS1 operations for vectors . . . 165

checkmat.c
Check matrix properties . . . 170
coarsening_cr.c
Coarsening with Brannick-Falgout strategy . . . 174
coarsening_rs.c
Coarsening with a modified Ruge-Stuben strategy . . . 175
convert.c
Some utilities for format conversion . . . 176
dlmalloc.h . . . ??
doxygen.h
Main page for Doygen documentation . . . 180
eigen.c
Subroutines for computing the extreme eigenvalues . . . 180
famg.c
Full AMG method as an iterative solver (main file) . . . 181
fasp.h
Main header file for FASP . . . 182
fasp_block.h
Header file for FASP block matrices . . . 194
fasp_const.h
Definition of all kinds of messages, including error messages, solver types, etc . . . 197
fmgcycle.c
Abstract non-recursive full multigrid cycle . . . 226
formats.c
Subroutines for matrix format conversion . . . 227
givens.c
Givens transformation . . . 233
gmg_poisson.c
GMG method as an iterative solver for Poisson Problem . . . 234
graphics.c
Subroutines for graphical output . . . 243
hb_io.h . . . ??
ilu.c
Incomplete LU decomposition: ILUk, ILUt, ILUtp . . . 247
ilu_setup_bsr.c
Setup incomplete LU decomposition for dBSRmat matrices . . . 255
ilu_setup_csr.c
Setup incomplete LU decomposition for dCSRmat matrices . . . 259
ilu_setup_str.c
Setup incomplete LU decomposition for dSTRmat matrices . . . 260
init.c
Initialize important data structures . . . 262
input.c
Read input parameters . . . 267
interface_mumps.c
Interface to MUMPS direct solvers . . . 269
interface_pardiso.c
Interface to Intel MKL PARDISO direct solvers . . . 272
interface_samg.c
Interface to SAMG solvers . . . 273
interface_superlu.c
Interface to SuperLU direct solvers . . . 275
interface_umfpack.c
Interface to UMFPACK direct solvers . . . 276

interpolation.c
Interpolation operators for AMG . . . 277
interpolation_em.c
Interpolation operators for AMG based on energy-min . . . 280
io.c
Matrix/vector input/output subroutines . . . 281
itsolver_blc.c
Iterative solvers for dBLCmat matrices . . . 309
itsolver_bsr.c
Iterative solvers for dBSRmat matrices . . . 313
itsolver_csr.c
Iterative solvers for dCSRmat matrices . . . 320
itsolver_mf.c
Iterative solvers using matrix-free spmv operations . . . 327
itsolver_str.c
Iterative solvers for dSTRmat matrices . . . 330
lu.c
LU decomposition and direct solver for small dense matrices . . . 334
malloc.c.h . . . ??
memory.c
Memory allocation and deallocation subroutines . . . 336
message.c
Output some useful messages . . . 341
mgcycle.c
Abstract multigrid cycle – non-recursive version . . . 345
mgrecur.c
Abstract multigrid cycle – recursive version . . . 346
nedmalloc.h . . . ??
ordering.c
Subroutines for ordering, merging, removing duplicated integers . . . 348
parameters.c
Initialize, set, or print input data and parameters . . . 356
pbcgs.c
Krylov subspace methods – Preconditioned BiCGstab . . . 368
pbcgs_mf.c
Krylov subspace methods – Preconditioned BiCGstab (matrix free) . . . 376
pcg.c
Krylov subspace methods – Preconditioned conjugate gradient . . . 380
pcg_mf.c
Krylov subspace methods – Preconditioned conjugate gradient (matrix free) . . . 385
pgcg.c
Krylov subspace methods – Preconditioned Generalized CG . . . 387
pgcg_mf.c
Krylov subspace methods – Preconditioned Generalized CG (matrix free) . . . 388
pgcr.c
Krylov subspace methods – Preconditioned GCR . . . 390
pgmres.c
Krylov subspace methods – Right-preconditioned GMRes . . . 393
pgmres_mf.c
Krylov subspace methods – Preconditioned GMRes (matrix free) . . . 398
pminres.c
Krylov subspace methods – Preconditioned minimal residual . . . 399
pminres_mf.c
Krylov subspace methods – Preconditioned minimal residual (matrix free) . . . 403

precond_blc.c
Preconditioners for dBLCmat matrices 406
precond_bsr.c
Preconditioners for dBSRmat matrices 416
precond_csr.c
Preconditioners for dCSRmat matrices 426
precond_str.c
Preconditioners for dSTRmat matrices 435
pvfgmres.c
Krylov subspace methods – Preconditioned variable-restarting flexible GMRes 442
pvfgmres_mf.c
Krylov subspace methods – Preconditioned variable-restarting flexible GMRes (matrix free) 446
pvgmres.c
Krylov subspace methods – Preconditioned variable-restart GMRes 447
pvgmres_mf.c
Krylov subspace methods – Preconditioned variable-restarting GMRes (matrix free) 451
quadrature.c
Quadrature rules 453
rap.c
Tripple-matrix multiplication R∗A∗P 455
schwarz_setup.c
Setup phase for the Schwarz methods 456
smat.c
Simple operations for *small* dense matrices in row-major format 459
smoother_bsr.c
Smoothers for dBSRmat matrices 468
smoother_csr.c
Smoothers for dCSRmat matrices 481
smoother_csr_cr.c
Smoothers for dCSRmat matrices using compatible relaxation 489
smoother_csr_poly.c
Smoothers for dCSRmat matrices using poly. approx. to A^{-1} 490
smoother_str.c
Smoothers for dSTRmat matrices 493
sparse_block.c
Sparse matrix block operations 506
sparse_bsr.c
Sparse matrix operations for dBSRmat matrices 510
sparse_coo.c
Sparse matrix operations for dCOOmat matrices 522
sparse_csr.c
Sparse matrix operations for dCSRmat matrices 525
sparse_csrl.c
Sparse matrix operations for dCSRLmat matrices 543
sparse_str.c
Sparse matrix operations for dSTRmat matrices 545
sparse_util.c
Routines for sparse matrix operations 549
spbcgs.c
Krylov subspace methods – Preconditioned BiCGstab with safety net 559
spcg.c
Krylov subspace methods – Preconditioned conjugate gradient with safety net 565
spgmres.c
Krylov subspace methods – Preconditioned GMRes with safety net 569

spminres.c
Krylov subspace methods – Preconditioned minimal residual with safety net 573
spvgmres.c
Krylov subspace methods – Preconditioned variable-restart GMRes with safety net 578
threads.c
Get and set number of threads and assign work load for each thread 583
timing.c
Timing subroutines 585
vec.c
Simple operations for vectors 586
wrapper.c
Wrappers for accessing functions by advanced users 595

Chapter 9

Data Structure Documentation

9.1 AMG_data Struct Reference

Data for AMG solvers.

```
#include <fasp.h>
```

Data Fields

- SHORT max_levels

  max number of levels
- SHORT num_levels

  number of levels in use <= max_levels
- dCSRmat A

  pointer to the matrix at level level_num
- dCSRmat R

  restriction operator at level level_num
- dCSRmat P

  prolongation operator at level level_num
- dvector b

  pointer to the right-hand side at level level_num
- dvector x

  pointer to the iterative solution at level level_num
- void $*$ Numeric

  pointer to the numerical factorization from UMFPACK
- Pardiso_data pdata

  data for Intel MKL PARDISO
- ivector cfmark

  pointer to the CF marker at level level_num
- INT ILU_levels

  number of levels use ILU smoother

- ILU_data LU

  ILU matrix for ILU smoother.
- INT near_kernel_dim

  dimension of the near kernel for SAMG
- REAL ** near_kernel_basis

  basis of near kernel space for SAMG
- INT Schwarz_levels

  number of levels use Schwarz smoother
- Schwarz_data Schwarz

  data of Schwarz smoother
- dvector w

  temporary work space
- Mumps_data mumps

  data for MUMPS
- INT cycle_type

  cycle type
- INT * ic

  indices for different colors
- INT * icmap

  mapping from vertex to color
- INT colors

  number of colors
- REAL weight

  weight for smoother

9.1.1 Detailed Description

Data for AMG solvers.

Note

This is needed for the AMG solver/preconditioner.

Definition at line 757 of file fasp.h.

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

- fasp.h

9.2 AMG_data_bsr Struct Reference

Data for multigrid levels. (BSR format)

```
#include <fasp_block.h>
```

Data Fields

- INT max_levels

    max number of levels
- INT num_levels

    number of levels in use <= max_levels
- dBSRmat A

    pointer to the matrix at level level_num
- dBSRmat R

    restriction operator at level level_num
- dBSRmat P

    prolongation operator at level level_num
- dvector b

    pointer to the right-hand side at level level_num
- dvector x

    pointer to the iterative solution at level level_num
- dvector diaginv

    pointer to the diagonal inverse at level level_num
- dCSRmat Ac

    pointer to the matrix at level level_num (csr format)
- void * Numeric

    pointer to the numerical dactorization from UMFPACK
- Pardiso_data pdata

    data for Intel MKL PARDISO
- dCSRmat PP

    pointer to the pressure block (only for reservoir simulation)
- REAL * pw

    pointer to the auxiliary vectors for pressure block
- dBSRmat SS

    pointer to the saturation block (only for reservoir simulation)
- REAL * sw

    pointer to the auxiliary vectors for saturation block
- dvector diaginv_SS

    pointer to the diagonal inverse of the saturation block at level level_num
- ILU_data PP_LU

    ILU data for pressure block.
- ivector cfmark

    pointer to the CF marker at level level_num
- INT ILU_levels

    number of levels use ILU smoother
- ILU_data LU

    ILU matrix for ILU smoother.
- INT near_kernel_dim

    dimension of the near kernel for SAMG
- REAL ** near_kernel_basis

    basis of near kernel space for SAMG
- dCSRmat * A_nk

Matrix data for near kernal.

- dCSRmat ∗ P_nk

  Prolongation for near kernal.

- dCSRmat ∗ R_nk

  Resriction for near kernal.

- dvector w

  temporary work space

- Mumps_data mumps

  data for MUMPS

9.2.1 Detailed Description

Data for multigrid levels. (BSR format)

Note

This structure is needed for the AMG solver/preconditioner in BSR format

Definition at line 198 of file fasp_block.h.

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

- fasp_block.h

9.3 AMG_param Struct Reference

Parameters for AMG solver.

```
#include <fasp.h>
```

Data Fields

- SHORT AMG_type

  type of AMG method

- SHORT print_level

  print level for AMG

- INT maxit

  max number of iterations of AMG

- REAL tol

  stopping tolerance for AMG solver

- SHORT max_levels

  max number of levels of AMG

- INT coarse_dof

  max number of coarsest level DOF

- SHORT cycle_type

  type of AMG cycle
- REAL quality_bound

  quality threshold for pairwise aggregation
- SHORT smoother

  smoother type
- SHORT smooth_order

  smoother order
- SHORT presmooth_iter

  number of presmoothers
- SHORT postsmooth_iter

  number of postsmoothers
- REAL relaxation

  relaxation parameter for SOR smoother
- SHORT polynomial_degree

  degree of the polynomial smoother
- SHORT coarse_solver

  coarse solver type
- SHORT coarse_scaling

  switch of scaling of the coarse grid correction
- SHORT amli_degree

  degree of the polynomial used by AMLI cycle
- REAL $*$ amli_coef

  coefficients of the polynomial used by AMLI cycle
- SHORT nl_amli_krylov_type

  type of Krylov method used by Nonlinear AMLI cycle
- SHORT coarsening_type

  coarsening type
- SHORT aggregation_type

  aggregation type
- SHORT interpolation_type

  interpolation type
- REAL strong_threshold

  strong connection threshold for coarsening
- REAL max_row_sum

  maximal row sum parameter
- REAL truncation_threshold

  truncation threshold
- INT aggressive_level

  number of levels use aggressive coarsening
- INT aggressive_path

  number of paths use to determine strongly coupled C points
- INT pair_number

  number of pairwise matchings
- REAL strong_coupled

  strong coupled threshold for aggregate
- INT max_aggregation

max size of each aggregate
- REAL tentative_smooth

  relaxation parameter for smoothing the tentative prolongation
- SHORT smooth_filter

  switch for filtered matrix used for smoothing the tentative prolongation
- SHORT ILU_levels

  number of levels use ILU smoother
- SHORT ILU_type

  ILU type for smoothing.
- INT ILU_lfil

  level of fill-in for ILUs and ILUk
- REAL ILU_droptol

  drop tolerance for ILUt
- REAL ILU_relax

  relaxation for ILUs
- REAL ILU_permtol

  permuted if permtol$*|a(i,j)| > |a(i,i)|$
- INT Schwarz_levels

  number of levels use Schwarz smoother
- INT Schwarz_mmsize

  maximal block size
- INT Schwarz_maxlvl

  maximal levels
- INT Schwarz_type

  type of Schwarz method
- INT Schwarz_blksolver

  type of Schwarz block solver

9.3.1 Detailed Description

Parameters for AMG solver.

Note

This is needed for the AMG solver/preconditioner.

Definition at line 618 of file fasp.h.

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

- fasp.h

9.4 block_BSR Struct Reference

Block REAL matrix format for reservoir simulation.

```
#include <fasp_block.h>
```

Data Fields

- dBSRmat ResRes

  reservoir-reservoir block
- dCSRmat ResWel

  reservoir-well block
- dCSRmat WelRes

  well-reservoir block
- dCSRmat WelWel

  well-well block

9.4.1 Detailed Description

Block REAL matrix format for reservoir simulation.

Definition at line 172 of file fasp_block.h.

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

- fasp_block.h

9.5 block_dvector Struct Reference

Block REAL vector structure.

```
#include <fasp_block.h>
```

Data Fields

- INT brow

  row number of blocks in A, m
- dvector ** blocks

  blocks of dvector, point to blocks[brow]

9.5.1 Detailed Description

Block REAL vector structure.

Definition at line 120 of file fasp_block.h.

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

- fasp_block.h

9.6 block_iCSRmat Struct Reference

Block INT CSR matrix format.

`#include <fasp_block.h>`

Data Fields

- INT brow

  row number of blocks in A, m
- INT bcol

  column number of blocks A, n
- iCSRmat ** blocks

  blocks of iCSRmat, point to blocks[brow][bcol]

9.6.1 Detailed Description

Block INT CSR matrix format.

Note

The starting index of A is 0.

Definition at line 103 of file fasp_block.h.

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

- fasp_block.h

9.7 block_ivector Struct Reference

Block INT vector structure.

`#include <fasp_block.h>`

Data Fields

- INT brow

  row number of blocks in A, m
- ivector ** blocks

  blocks of dvector, point to blocks[brow]

9.7.1 Detailed Description

Block INT vector structure.

Note

The starting index of A is 0.

Definition at line 136 of file fasp_block.h.

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

- fasp_block.h

9.8 block_Reservoir Struct Reference

Block REAL matrix format for reservoir simulation.

```
#include <fasp_block.h>
```

Data Fields

- dSTRmat ResRes

  reservoir-reservoir block
- dCSRmat ResWel

  reservoir-well block
- dCSRmat WelRes

  well-reservoir block
- dCSRmat WelWel

  well-well block

9.8.1 Detailed Description

Block REAL matrix format for reservoir simulation.

Definition at line 151 of file fasp_block.h.

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

- fasp_block.h

9.9 dBLCmat Struct Reference

Block REAL CSR matrix format.

```
#include <fasp_block.h>
```

Data Fields

- INT brow

  row number of blocks in A, m
- INT bcol

  column number of blocks A, n
- dCSRmat ** blocks

  blocks of dCSRmat, point to blocks[brow][bcol]

9.9.1 Detailed Description

Block REAL CSR matrix format.

Note

The starting index of A is 0.

Definition at line 84 of file fasp_block.h.

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

- fasp_block.h

9.10 dBSRmat Struct Reference

Block sparse row storage matrix of REAL type.

```
#include <fasp_block.h>
```

Data Fields

- INT ROW

  number of rows of sub-blocks in matrix A, M
- INT COL

  number of cols of sub-blocks in matrix A, N
- INT NNZ

  number of nonzero sub-blocks in matrix A, NNZ
- INT nb

  dimension of each sub-block
- INT storage_manner

  storage manner for each sub-block
- REAL * val
- INT * IA

  integer array of row pointers, the size is ROW+1
- INT * JA

9.10.1 Detailed Description

Block sparse row storage matrix of REAL type.

Note

This data structure is adapted from the Intel MKL library. Refer to: http://software.intel.com/sites/products/documentation/hpc/mkl/lin/index.htm
Some of the following entries are capitalized to stress that they are for blocks!

Definition at line 44 of file fasp_block.h.

9.10.2 Field Documentation

9.10.2.1 JA

```
INT* JA
```

Element i of the integer array columns is the number of the column in the block matrix that contains the i-th non-zero block. The size is NNZ.

Definition at line 74 of file fasp_block.h.

9.10.2.2 val

```
REAL* val
```

A real array that contains the elements of the non-zero blocks of a sparse matrix. The elements are stored block-by-block in row major order. A non-zero block is the block that contains at least one non-zero element. All elements of non-zero blocks are stored, even if some of them is equal to zero. Within each nonzero block elements are stored in row-major order and the size is (NNZ∗nb∗nb).

Definition at line 67 of file fasp_block.h.

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

- fasp_block.h

9.11 dCOOmat Struct Reference

Sparse matrix of REAL type in COO (or IJ) format.

```
#include <fasp.h>
```

Data Fields

- INT row
  row number of matrix A, m
- INT col
  column of matrix A, n
- INT nnz
  number of nonzero entries
- INT * rowind
  integer array of row indices, the size is nnz
- INT * colind
  integer array of column indices, the size is nnz
- REAL * val
  nonzero entries of A

9.11.1 Detailed Description

Sparse matrix of REAL type in COO (or IJ) format.

Coordinate Format (I,J,A)

Note

The starting index of A is 0.
Change I to rowind, J to colind. To avoid with complex.h confliction on I.

Definition at line 214 of file fasp.h.

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

- fasp.h

9.12 dCSRLmat Struct Reference

Sparse matrix of REAL type in CSRL format.

```
#include <fasp.h>
```

Data Fields

- INT row
  number of rows
- INT col
  number of cols
- INT nnz
  number of nonzero entries
- INT dif
  number of different values in i-th row, i=0:nrows-1
- INT ∗ nz_diff
  nz_diff[i]: the i-th different value in 'nzrow'
- INT ∗ index
  row index of the matrix (length-grouped): rows with same nnz are together
- INT ∗ start
  j in {start[i],...,start[i+1]-1} means nz_diff[i] nnz in index[j]-row
- INT ∗ ja
  column indices of all the nonzeros
- REAL ∗ val
  values of all the nonzero entries

9.12.1 Detailed Description

Sparse matrix of REAL type in CSRL format.

Definition at line 270 of file fasp.h.

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

- fasp.h

9.13 dCSRmat Struct Reference

Sparse matrix of REAL type in CSR format.

```
#include <fasp.h>
```

Data Fields

- INT row
  row number of matrix A, m
- INT col
  column of matrix A, n
- INT nnz
  number of nonzero entries
- INT ∗ IA
  integer array of row pointers, the size is m+1
- INT ∗ JA
  integer array of column indexes, the size is nnz
- REAL ∗ val
  nonzero entries of A

9.13.1 Detailed Description

Sparse matrix of REAL type in CSR format.

CSR Format (IA,JA,A) in REAL

Note

The starting index of A is 0.

Definition at line 153 of file fasp.h.

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

- fasp.h

9.14 ddenmat Struct Reference

Dense matrix of REAL type.

`#include <fasp.h>`

Data Fields

- INT row
  number of rows
- INT col
  number of columns
- REAL ** val
  actual matrix entries

9.14.1 Detailed Description

Dense matrix of REAL type.

A dense REAL matrix

Definition at line 113 of file fasp.h.

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

- fasp.h

9.15 dSTRmat Struct Reference

Structure matrix of REAL type.

```
#include <fasp.h>
```

Data Fields

- INT nx

  number of grids in x direction
- INT ny

  number of grids in y direction
- INT nz

  number of grids in z direction
- INT nxy

  number of grids on x-y plane
- INT nc

  size of each block (number of components)
- INT ngrid

  number of grids
- REAL ∗ diag

  diagonal entries (length is ngrid∗(nc^2))
- INT nband

  number of off-diag bands
- INT ∗ offsets

  offsets of the off-diagonals (length is nband)
- REAL ∗∗ offdiag

  off-diagonal entries (dimension is nband ∗ [(ngrid-|offsets|) ∗ nc^2])

9.15.1 Detailed Description

Structure matrix of REAL type.

Note

Every nc^2 entries of the array diag and off-diag[i] store one block: For 2D matrix, the recommended offsets is [-1,1,-nx,nx]; For 3D matrix, the recommended offsets is [-1,1,-nx,nx,-nxy,nxy].

Definition at line 309 of file fasp.h.

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

- fasp.h

9.16 dvector Struct Reference

Vector with n entries of REAL type.

```
#include <fasp.h>
```

Data Fields

- INT row
  number of rows
- REAL ∗ val
  actual vector entries

9.16.1 Detailed Description

Vector with n entries of REAL type.

Definition at line 347 of file fasp.h.

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

- fasp.h

9.17 grid2d Struct Reference

Two dimensional grid data structure.

```
#include <fasp.h>
```

Data Fields

- REAL(∗ p)[2]
- INT(∗ e)[2]
- INT(∗ t)[3]
- INT(∗ s)[3]
- INT ∗ pdiri
- INT ∗ ediri
- INT ∗ pfather
- INT ∗ efather
- INT ∗ tfather
- INT vertices
- INT edges
- INT triangles

9.17.1 Detailed Description

Two dimensional grid data structure.

Note

The grid2d structure is simply a list of triangles, edges and vertices. edge i has 2 vertices e[i], triangle i has 3 edges s[i], 3 vertices t[i] vertex i has two coordinates p[i]

Definition at line 1174 of file fasp.h.

9.17.2 Field Documentation

9.17.2.1 e

```
INT(* e)[2]
```

Vertices of edges

Definition at line 1177 of file fasp.h.

9.17.2.2 edges

```
INT edges
```

Number of edges

Definition at line 1188 of file fasp.h.

9.17.2.3 ediri

```
INT* ediri
```

Boundary flags (0 <=> interior edge)

Definition at line 1181 of file fasp.h.

9.17.2.4 efather

```
INT* efather
```

Father edge or triangle

Definition at line 1184 of file fasp.h.

9.17.2.5 p

```
REAL(* p)[2]
```

Coordinates of vertices

Definition at line 1176 of file fasp.h.

9.17.2.6 pdiri

```
INT* pdiri
```

Boundary flags (0 <=> interior point)

Definition at line 1180 of file fasp.h.

9.17.2.7 pfather

```
INT* pfather
```

Father point or edge

Definition at line 1183 of file fasp.h.

9.17.2.8 s

```
INT(* s)[3]
```

Edges of triangles

Definition at line 1179 of file fasp.h.

9.17.2.9 t

```
INT(* t)[3]
```

Vertices of triangles

Definition at line 1178 of file fasp.h.

9.17.2.10 tfather

```
INT* tfather
```

Father triangle

Definition at line 1185 of file fasp.h.

9.17.2.11 triangles

```
INT triangles
```

Number of triangles

Definition at line 1189 of file fasp.h.

9.17.2.12 vertices

```
INT vertices
```

Number of grid points

Definition at line 1187 of file fasp.h.

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

- fasp.h

9.18 iCOOmat Struct Reference

Sparse matrix of INT type in COO (or IJ) format.

```
#include <fasp.h>
```

Data Fields

- INT row

  row number of matrix A, m
- INT col

  column of matrix A, n
- INT nnz

  number of nonzero entries
- INT ∗ I

  integer array of row indices, the size is nnz
- INT ∗ J

  integer array of column indices, the size is nnz
- INT ∗ val

  nonzero entries of A

9.18.1 Detailed Description

Sparse matrix of INT type in COO (or IJ) format.

Coordinate Format (I,J,A)

Note

The starting index of A is 0.

Definition at line 244 of file fasp.h.

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

- fasp.h

9.19 iCSRmat Struct Reference

Sparse matrix of INT type in CSR format.

```
#include <fasp.h>
```

Data Fields

- INT row
  row number of matrix A, m
- INT col
  column of matrix A, n
- INT nnz
  number of nonzero entries
- INT ∗ IA
  integer array of row pointers, the size is m+1
- INT ∗ JA
  integer array of column indexes, the size is nnz
- INT ∗ val
  nonzero entries of A

9.19.1 Detailed Description

Sparse matrix of INT type in CSR format.

CSR Format (IA,JA,A) in integer

Note

The starting index of A is 0.

Definition at line 183 of file fasp.h.

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

- fasp.h

9.20 idenmat Struct Reference

Dense matrix of INT type.

```
#include <fasp.h>
```

Data Fields

- INT row

  number of rows
- INT col

  number of columns
- INT ** val

  actual matrix entries

9.20.1 Detailed Description

Dense matrix of INT type.

A dense INT matrix

Definition at line 132 of file fasp.h.

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

- fasp.h

9.21 ILU_data Struct Reference

Data for ILU setup.

```
#include <fasp.h>
```

Data Fields

- INT row
  row number of matrix LU, m
- INT col
  column of matrix LU, n
- INT nzlu
  number of nonzero entries
- INT ∗ ijlu
  integer array of row pointers and column indexes, the size is nzlu
- REAL ∗ luval
  nonzero entries of LU
- INT nb
  block size for BSR type only
- INT nwork
  work space size
- REAL ∗ work
  work space
- INT ncolors
  number of colors for multi-threading
- INT ∗ ic
  indices for different colors
- INT ∗ icmap
  mapping from vertex to color
- INT ∗ uptr
  temporary work space
- INT nlevL
  number of colors for lower triangle
- INT nlevU
  number of colors for upper triangle
- INT ∗ ilevL
  number of vertices in each color for lower triangle
- INT ∗ ilevU
  number of vertices in each color for upper triangle
- INT ∗ jlevL
  mapping from row to color for lower triangle
- INT ∗ jlevU
  mapping from row to color for upper triangle

9.21.1 Detailed Description

Data for ILU setup.

Definition at line 405 of file fasp.h.

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

- fasp.h

9.22 ILU_param Struct Reference

Parameters for ILU.

`#include <fasp.h>`

Data Fields

- SHORT print_level

  print level
- SHORT ILU_type

  ILU type for decomposition.
- INT ILU_lfil

  level of fill-in for ILUk
- REAL ILU_droptol

  drop tolerance for ILUt
- REAL ILU_relax

  add the sum of dropped elements to diagonal element in proportion relax
- REAL ILU_permtol

  permuted if permtol$*|a(i,j)| > |a(i,i)|$

9.22.1 Detailed Description

Parameters for ILU.

Definition at line 379 of file fasp.h.

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

- fasp.h

9.23 input_param Struct Reference

Input parameters.

`#include <fasp.h>`

Data Fields

- SHORT print_level
- SHORT output_type
- char inifile [256]
- char workdir [256]
- INT problem_num
- SHORT solver_type
- SHORT precond_type
- SHORT stop_type
- REAL itsolver_tol
- INT itsolver_maxit
- INT restart
- SHORT ILU_type
- INT ILU_lfil
- REAL ILU_droptol
- REAL ILU_relax
- REAL ILU_permtol
- INT Schwarz_mmsize
- INT Schwarz_maxlvl
- INT Schwarz_type
- INT Schwarz_blksolver
- SHORT AMG_type
- SHORT AMG_levels
- SHORT AMG_cycle_type
- SHORT AMG_smoother
- SHORT AMG_smooth_order
- REAL AMG_relaxation
- SHORT AMG_polynomial_degree
- SHORT AMG_presmooth_iter
- SHORT AMG_postsmooth_iter
- INT AMG_coarse_dof
- REAL AMG_tol
- INT AMG_maxit
- SHORT AMG_ILU_levels
- SHORT AMG_coarse_solver
- SHORT AMG_coarse_scaling
- SHORT AMG_amli_degree
- SHORT AMG_nl_amli_krylov_type
- INT AMG_Schwarz_levels
- SHORT AMG_coarsening_type
- SHORT AMG_aggregation_type
- SHORT AMG_interpolation_type
- REAL AMG_strong_threshold
- REAL AMG_truncation_threshold
- REAL AMG_max_row_sum
- INT AMG_aggressive_level
- INT AMG_aggressive_path
- INT AMG_pair_number
- REAL AMG_quality_bound
- REAL AMG_strong_coupled
- INT AMG_max_aggregation
- REAL AMG_tentative_smooth
- SHORT AMG_smooth_filter

9.23.1 Detailed Description

Input parameters.

Input parameters, reading from disk file

Definition at line 1076 of file fasp.h.

9.23.2 Field Documentation

9.23.2.1 AMG_aggregation_type

```
SHORT AMG_aggregation_type
```

aggregation type

Definition at line 1130 of file fasp.h.

9.23.2.2 AMG_aggressive_level

```
INT AMG_aggressive_level
```

number of levels use aggressive coarsening

Definition at line 1135 of file fasp.h.

9.23.2.3 AMG_aggressive_path

```
INT AMG_aggressive_path
```

number of paths used to determine strongly coupled C-set

Definition at line 1136 of file fasp.h.

9.23.2.4 AMG_amli_degree

```
SHORT AMG_amli_degree
```

degree of the polynomial used by AMLI cycle

Definition at line 1124 of file fasp.h.

9.23.2.5 AMG_coarse_dof

`INT AMG_coarse_dof`

max number of coarsest level DOF

Definition at line 1118 of file fasp.h.

9.23.2.6 AMG_coarse_scaling

`SHORT AMG_coarse_scaling`

switch of scaling of the coarse grid correction

Definition at line 1123 of file fasp.h.

9.23.2.7 AMG_coarse_solver

`SHORT AMG_coarse_solver`

coarse solver type

Definition at line 1122 of file fasp.h.

9.23.2.8 AMG_coarsening_type

`SHORT AMG_coarsening_type`

coarsening type

Definition at line 1129 of file fasp.h.

9.23.2.9 AMG_cycle_type

`SHORT AMG_cycle_type`

type of cycle

Definition at line 1111 of file fasp.h.

9.23.2.10 AMG_ILU_levels

`SHORT AMG_ILU_levels`

how many levels use ILU smoother

Definition at line 1121 of file fasp.h.

9.23.2.11 AMG_interpolation_type

```
SHORT AMG_interpolation_type
```

interpolation type

Definition at line 1131 of file fasp.h.

9.23.2.12 AMG_levels

```
SHORT AMG_levels
```

maximal number of levels

Definition at line 1110 of file fasp.h.

9.23.2.13 AMG_max_aggregation

```
INT AMG_max_aggregation
```

max size of each aggregate

Definition at line 1142 of file fasp.h.

9.23.2.14 AMG_max_row_sum

```
REAL AMG_max_row_sum
```

maximal row sum

Definition at line 1134 of file fasp.h.

9.23.2.15 AMG_maxit

```
INT AMG_maxit
```

number of iterations for AMG used as preconditioner

Definition at line 1120 of file fasp.h.

9.23.2.16 AMG_nl_amli_krylov_type

```
SHORT AMG_nl_amli_krylov_type
```

type of Krylov method used by nonlinear AMLI cycle

Definition at line 1125 of file fasp.h.

9.23.2.17 AMG_pair_number

```
INT AMG_pair_number
```

number of pairs in matching algorithm

Definition at line 1137 of file fasp.h.

9.23.2.18 AMG_polynomial_degree

```
SHORT AMG_polynomial_degree
```

degree of the polynomial smoother

Definition at line 1115 of file fasp.h.

9.23.2.19 AMG_postsmooth_iter

```
SHORT AMG_postsmooth_iter
```

number of postsmoothing

Definition at line 1117 of file fasp.h.

9.23.2.20 AMG_presmooth_iter

```
SHORT AMG_presmooth_iter
```

number of presmoothing

Definition at line 1116 of file fasp.h.

9.23.2.21 AMG_quality_bound

```
REAL AMG_quality_bound
```

threshold for pair wise aggregation

Definition at line 1138 of file fasp.h.

9.23.2.22 AMG_relaxation

```
REAL AMG_relaxation
```

over-relaxation parameter for SOR

Definition at line 1114 of file fasp.h.

9.23.2.23 AMG_Schwarz_levels

`INT AMG_Schwarz_levels`

number of levels use Schwarz smoother

Definition at line 1126 of file fasp.h.

9.23.2.24 AMG_smooth_filter

`SHORT AMG_smooth_filter`

use filter for smoothing the tentative prolongation or not

Definition at line 1144 of file fasp.h.

9.23.2.25 AMG_smooth_order

`SHORT AMG_smooth_order`

order for smoothers

Definition at line 1113 of file fasp.h.

9.23.2.26 AMG_smoother

`SHORT AMG_smoother`

type of smoother

Definition at line 1112 of file fasp.h.

9.23.2.27 AMG_strong_coupled

`REAL AMG_strong_coupled`

strong coupled threshold for aggregate

Definition at line 1141 of file fasp.h.

9.23.2.28 AMG_strong_threshold

`REAL AMG_strong_threshold`

strong threshold for coarsening

Definition at line 1132 of file fasp.h.

9.23.2.29 AMG_tentative_smooth

```
REAL AMG_tentative_smooth
```

relaxation factor for smoothing the tentative prolongation

Definition at line 1143 of file fasp.h.

9.23.2.30 AMG_tol

```
REAL AMG_tol
```

tolerance for AMG if used as preconditioner

Definition at line 1119 of file fasp.h.

9.23.2.31 AMG_truncation_threshold

```
REAL AMG_truncation_threshold
```

truncation factor for interpolation

Definition at line 1133 of file fasp.h.

9.23.2.32 AMG_type

```
SHORT AMG_type
```

Type of AMG

Definition at line 1109 of file fasp.h.

9.23.2.33 ILU_droptol

```
REAL ILU_droptol
```

drop tolerance

Definition at line 1098 of file fasp.h.

9.23.2.34 ILU_lfil

```
INT ILU_lfil
```

level of fill-in

Definition at line 1097 of file fasp.h.

9.23.2.35 ILU_permtol

```
REAL ILU_permtol
```

permutation tolerance

Definition at line 1100 of file fasp.h.

9.23.2.36 ILU_relax

```
REAL ILU_relax
```

scaling factor: add the sum of dropped entries to diagonal

Definition at line 1099 of file fasp.h.

9.23.2.37 ILU_type

```
SHORT ILU_type
```

ILU type for decomposition

Definition at line 1096 of file fasp.h.

9.23.2.38 inifile

```
char inifile[256]
```

ini file name

Definition at line 1083 of file fasp.h.

9.23.2.39 itsolver_maxit

```
INT itsolver_maxit
```

maximal number of iterations for iterative solvers

Definition at line 1092 of file fasp.h.

9.23.2.40 itsolver_tol

```
REAL itsolver_tol
```

tolerance for iterative linear solver

Definition at line 1091 of file fasp.h.

9.23.2.41 output_type

```
SHORT output_type
```

type of output stream

Definition at line 1080 of file fasp.h.

9.23.2.42 precond_type

```
SHORT precond_type
```

type of preconditioner for iterative solvers

Definition at line 1089 of file fasp.h.

9.23.2.43 print_level

```
SHORT print_level
```

print level

Definition at line 1079 of file fasp.h.

9.23.2.44 problem_num

```
INT problem_num
```

problem number to solve

Definition at line 1085 of file fasp.h.

9.23.2.45 restart

```
INT restart
```

restart number used in GMRES

Definition at line 1093 of file fasp.h.

9.23.2.46 Schwarz_blksolver

```
INT Schwarz_blksolver
```

type of Schwarz block solver

Definition at line 1106 of file fasp.h.

9.23.2.47 Schwarz_maxlvl

```
INT Schwarz_maxlvl
```

maximal levels

Definition at line 1104 of file fasp.h.

9.23.2.48 Schwarz_mmsize

```
INT Schwarz_mmsize
```

maximal block size

Definition at line 1103 of file fasp.h.

9.23.2.49 Schwarz_type

```
INT Schwarz_type
```

type of Schwarz method

Definition at line 1105 of file fasp.h.

9.23.2.50 solver_type

```
SHORT solver_type
```

type of iterative solvers

Definition at line 1088 of file fasp.h.

9.23.2.51 stop_type

```
SHORT stop_type
```

type of stopping criteria for iterative solvers

Definition at line 1090 of file fasp.h.

9.23.2.52 workdir

```
char workdir[256]
```

working directory for data files

Definition at line 1084 of file fasp.h.

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

- fasp.h

9.24 itsolver_param Struct Reference

Parameters passed to iterative solvers.

```
#include <fasp.h>
```

Data Fields

- SHORT itsolver_type
- SHORT precond_type
- SHORT stop_type
- INT maxit
- REAL tol
- INT restart
- SHORT print_level

9.24.1 Detailed Description

Parameters passed to iterative solvers.

Definition at line 1152 of file fasp.h.

9.24.2 Field Documentation

9.24.2.1 itsolver_type

```
SHORT itsolver_type
```

solver type: see message.h

Definition at line 1154 of file fasp.h.

9.24.2.2 maxit

```
INT maxit
```

max number of iterations

Definition at line 1157 of file fasp.h.

9.24.2.3 precond_type

```
SHORT precond_type
```

preconditioner type: see message.h

Definition at line 1155 of file fasp.h.

9.24.2.4 print_level

```
SHORT print_level
```

print level: 0–10

Definition at line 1160 of file fasp.h.

9.24.2.5 restart

```
INT restart
```

number of steps for restarting: for GMRES etc

Definition at line 1159 of file fasp.h.

9.24.2.6 stop_type

```
SHORT stop_type
```

stopping criteria type

Definition at line 1156 of file fasp.h.

9.24.2.7 tol

```
REAL tol
```

convergence tolerance

Definition at line 1158 of file fasp.h.

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

- fasp.h

9.25 ivector Struct Reference

Vector with n entries of INT type.

```
#include <fasp.h>
```

Data Fields

- INT row
  number of rows
- INT * val
  actual vector entries

9.25.1 Detailed Description

Vector with n entries of INT type.

Definition at line 361 of file fasp.h.

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

- fasp.h

9.26 Link Struct Reference

Struct for Links.

```
#include <fasp.h>
```

Data Fields

- INT prev
  previous node in the linklist
- INT next
  next node in the linklist

9.26.1 Detailed Description

Struct for Links.

Definition at line 1201 of file fasp.h.

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

- fasp.h

9.27 linked_list Struct Reference

A linked list node.

```
#include <fasp.h>
```

Data Fields

- INT data
  data
- INT head
  starting of the list
- INT tail
  ending of the list
- struct linked_list ∗ next_node
  next node
- struct linked_list ∗ prev_node
  previous node

9.27.1 Detailed Description

A linked list node.

Note

This definition is adapted from hypre 2.0.

Definition at line 1218 of file fasp.h.

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

- fasp.h

9.28 mallinfo Struct Reference

Data Fields

- MALLINFO_FIELD_TYPE **arena**
- MALLINFO_FIELD_TYPE **ordblks**
- MALLINFO_FIELD_TYPE **smblks**
- MALLINFO_FIELD_TYPE **hblks**
- MALLINFO_FIELD_TYPE **hblkhd**
- MALLINFO_FIELD_TYPE **usmblks**
- MALLINFO_FIELD_TYPE **fsmblks**
- MALLINFO_FIELD_TYPE **uordblks**
- MALLINFO_FIELD_TYPE **fordblks**
- MALLINFO_FIELD_TYPE **keepcost**

9.28.1 Detailed Description

Definition at line 69 of file dlmalloc.h.

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

- dlmalloc.h
- malloc.c.h

9.29 malloc_chunk Struct Reference

Data Fields

- size_t **prev_foot**
- size_t **head**
- struct malloc_chunk ∗ **fd**
- struct malloc_chunk ∗ **bk**

9.29.1 Detailed Description

Definition at line 2177 of file malloc.c.h.

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

- malloc.c.h

9.30 malloc_params Struct Reference

Data Fields

- volatile size_t **magic**
- size_t **page_size**
- size_t **granularity**
- size_t **mmap_threshold**
- size_t **trim_threshold**
- flag_t **default_mflags**

9.30.1 Detailed Description

Definition at line 1494 of file malloc.c.h.

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

- malloc.c.h

9.31 malloc_segment Struct Reference

Data Fields

- char ∗ **base**
- size_t **size**
- struct malloc_segment ∗ **next**
- flag_t **sflags**

9.31.1 Detailed Description

Definition at line 2458 of file malloc.c.h.

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

- malloc.c.h

9.32 malloc_state Struct Reference

Data Fields

- binmap_t **smallmap**
- binmap_t **treemap**
- size_t **dvsize**
- size_t **topsize**
- char * **least_addr**
- mchunkptr **dv**
- mchunkptr **top**
- size_t **trim_check**
- size_t **release_checks**
- size_t **magic**
- mchunkptr **smallbins** [(NSMALLBINS+1) *2]
- tbinptr **treebins** [NTREEBINS]
- size_t **footprint**
- size_t **max_footprint**
- flag_t **mflags**
- msegment **seg**
- void * **extp**
- size_t **exts**

9.32.1 Detailed Description

Definition at line 2565 of file malloc.c.h.

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

- malloc.c.h

9.33 malloc_tree_chunk Struct Reference

Data Fields

- size_t **prev_foot**
- size_t **head**
- struct malloc_tree_chunk * **fd**
- struct malloc_tree_chunk * **bk**
- struct malloc_tree_chunk * **child** [2]
- struct malloc_tree_chunk * **parent**
- bindex_t **index**

9.33.1 Detailed Description

Definition at line 2382 of file malloc.c.h.

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

- malloc.c.h

9.34 Mumps_data Struct Reference

Parameters for MUMPS interface.

```
#include <fasp.h>
```

Data Fields

- INT job
  work for MUMPS

9.34.1 Detailed Description

Parameters for MUMPS interface.

Added on 10/10/2014

Definition at line 494 of file fasp.h.

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

- fasp.h

9.35 mxv_matfree Struct Reference

Matrix-vector multiplication, replace the actual matrix.

```
#include <fasp.h>
```

Data Fields

- void ∗ data
  data for MxV, can be a Matrix or something else
- void(∗ fct)(void ∗, REAL ∗, REAL ∗)
  action for MxV, void function pointer

9.35.1 Detailed Description

Matrix-vector multiplication, replace the actual matrix.

Definition at line 1060 of file fasp.h.

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

- fasp.h

9.36 nedmallinfo Struct Reference

Data Fields

- size_t **arena**
- size_t **ordblks**
- size_t **smblks**
- size_t **hblks**
- size_t **hblkhd**
- size_t **usmblks**
- size_t **fsmblks**
- size_t **uordblks**
- size_t **fordblks**
- size_t **keepcost**

9.36.1 Detailed Description

Definition at line 168 of file nedmalloc.h.

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

- nedmalloc.h

9.37 Pardiso_data Struct Reference

Parameters for Intel MKL PARDISO interface.

```
#include <fasp.h>
```

Data Fields

- void ∗ pt [64]

    Internal solver memory pointer.

9.37.1 Detailed Description

Parameters for Intel MKL PARDISO interface.

Added on 11/28/2015

Definition at line 512 of file fasp.h.

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

- fasp.h

9.38 precond Struct Reference

Preconditioner data and action.

`#include <fasp.h>`

Data Fields

- void * data
  data for preconditioner, void pointer
- void(* fct)(REAL *, REAL *, void *)
  action for preconditioner, void function pointer

9.38.1 Detailed Description

Preconditioner data and action.

Note

This is the preconditioner structure for preconditioned iterative methods.

Definition at line 1046 of file fasp.h.

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

- fasp.h

9.39 precond_block_data Struct Reference

Data passed to the preconditioner for block preconditioning for dBLCmat format.

`#include <fasp_block.h>`

Data Fields

- dBLCmat * Ablc
- dCSRmat * A_diag
- dvector r
- void ** LU_diag
- AMG_data ** mgl
- AMG_param * amgparam

9.39.1 Detailed Description

Data passed to the preconditioner for block preconditioning for dBLCmat format.

This is needed for the block preconditioner.

Definition at line 502 of file fasp_block.h.

9.39.2 Field Documentation

9.39.2.1 A_diag

```
dCSRmat* A_diag
```

data for each diagonal block

Definition at line 509 of file fasp_block.h.

9.39.2.2 Ablc

```
dBLCmat* Ablc
```

problem data, the blocks

Definition at line 507 of file fasp_block.h.

9.39.2.3 amgparam

```
AMG_param* amgparam
```

parameters for AMG

Definition at line 521 of file fasp_block.h.

9.39.2.4 LU_diag

```
void** LU_diag
```

LU decomposition for the diagonal blocks (for UMFpack)

Definition at line 517 of file fasp_block.h.

9.39.2.5 mgl

```
AMG_data** mgl
```

AMG data for the diagonal blocks

Definition at line 520 of file fasp_block.h.

9.39.2.6 r

```
dvector r
```

temp work space

Definition at line 511 of file fasp_block.h.

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

- fasp_block.h

9.40 precond_block_reservoir_data Struct Reference

Data passed to the preconditioner for reservoir simulation problems.

```
#include <fasp_block.h>
```

Data Fields

- block_Reservoir ∗ A

  problem data in block_Reservoir format
- dBLCmat ∗ Ablc

  problem data in dBLCmat format
- dCSRmat ∗ Acsr

  problem data in CSR format
- INT ILU_lfil

  level of fill-in for structured ILU(k)
- dSTRmat ∗ LU

  LU matrix for Reservoir-Reservoir block in STR format.
- ILU_data ∗ LUcsr

  LU matrix for Reservoir-Reservoir block in CSR format.
- AMG_data ∗ mgl_data

  AMG data for presure-presure block.
- SHORT print_level

  print level in AMG preconditioner
- INT maxit_AMG

  max number of iterations of AMG preconditioner
- SHORT max_levels

  max number of AMG levels
- REAL amg_tol

  tolerance for AMG preconditioner
- SHORT cycle_type

  AMG cycle type.
- SHORT smoother

  AMG smoother type.
- SHORT presmooth_iter

  number of presmoothing
- SHORT postsmooth_iter

  number of postsmoothing
- SHORT coarsening_type

  coarsening type
- REAL relaxation

  relaxation parameter for SOR smoother
- SHORT coarse_scaling

  switch of scaling of coarse grid correction
- INT maxit

  max number of iterations
- INT restart

  number of iterations for restart
- REAL tol

  tolerance for convergence
- REAL ∗ invS

  inverse of the Schur complement (-I - Awr∗Arr^{-1}∗Arw)^{-1}, Arr may be replaced by LU
- dvector ∗ DPSinvDSS

Diag(PS) ∗ inv(Diag(SS))

- SHORT scaled
- ivector ∗ perf_idx
- dSTRmat ∗ RR
- dCSRmat ∗ WW
- dCSRmat ∗ PP
- dSTRmat ∗ SS
- precond_diagstr ∗ diag
- dvector ∗ diaginv
- ivector ∗ pivot
- dvector ∗ diaginvS
- ivector ∗ pivotS
- ivector ∗ order
- dvector r
- REAL ∗ w

9.40.1 Detailed Description

Data passed to the preconditioner for reservoir simulation problems.

Note

This is only needed for the Black Oil model with wells

Definition at line 404 of file fasp_block.h.

9.40.2 Field Documentation

9.40.2.1 diag

```
precond_diagstr* diag
```

the diagonal inverse for diagonal scaling

Definition at line 484 of file fasp_block.h.

9.40.2.2 diaginv

```
dvector* diaginv
```

the inverse of the diagonals for GS/block GS smoother (whole reservoir matrix)

Definition at line 485 of file fasp_block.h.

9.40.2.3 diaginvS

```
dvector* diaginvS
```

the inverse of the diagonals for GS/block GS smoother (saturation block)

Definition at line 487 of file fasp_block.h.

9.40.2.4 order

```
ivector* order
```

order for smoothing

Definition at line 489 of file fasp_block.h.

9.40.2.5 perf_idx

```
ivector* perf_idx
```

variable index for perf

Definition at line 477 of file fasp_block.h.

9.40.2.6 pivot

```
ivector* pivot
```

the pivot for the GS/block GS smoother (whole reservoir matrix)

Definition at line 486 of file fasp_block.h.

9.40.2.7 pivotS

```
ivector* pivotS
```

the pivot for the GS/block GS smoother (saturation block)

Definition at line 488 of file fasp_block.h.

9.40.2.8 PP

```
dCSRmat* PP
```

pressure block after diagonal scaling

Definition at line 481 of file fasp_block.h.

9.40.2.9 r

```
dvector r
```

temporary dvector used to store and restore the residual

Definition at line 492 of file fasp_block.h.

9.40.2.10 RR

```
dSTRmat* RR
```

Diagonal scaled reservoir block

Definition at line 479 of file fasp_block.h.

9.40.2.11 scaled

```
SHORT scaled
```

whether the matirx is scaled

Definition at line 476 of file fasp_block.h.

9.40.2.12 SS

```
dSTRmat* SS
```

saturation block after diaogonal scaling

Definition at line 482 of file fasp_block.h.

9.40.2.13 w

```
REAL* w
```

temporary work space for other usage

Definition at line 493 of file fasp_block.h.

9.40.2.14 WW

`dCSRmat* WW`

Argumented well block

Definition at line 480 of file fasp_block.h.

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

- fasp_block.h

9.41 precond_data Struct Reference

Data passed to the preconditioners.

`#include <fasp.h>`

Data Fields

- SHORT AMG_type
  type of AMG method
- SHORT print_level
  print level in AMG preconditioner
- INT maxit
  max number of iterations of AMG preconditioner
- SHORT max_levels
  max number of AMG levels
- REAL tol
  tolerance for AMG preconditioner
- SHORT cycle_type
  AMG cycle type.
- SHORT smoother
  AMG smoother type.
- SHORT smooth_order
  AMG smoother ordering.
- SHORT presmooth_iter
  number of presmoothing
- SHORT postsmooth_iter
  number of postsmoothing
- REAL relaxation
  relaxation parameter for SOR smoother
- SHORT polynomial_degree
  degree of the polynomial smoother
- SHORT coarsening_type

switch of scaling of the coarse grid correction

- SHORT coarse_solver

  coarse solver type for AMG
- SHORT coarse_scaling

  switch of scaling of the coarse grid correction
- SHORT amli_degree

  degree of the polynomial used by AMLI cycle
- SHORT nl_amli_krylov_type

  type of Krylov method used by Nonlinear AMLI cycle
- REAL tentative_smooth

  smooth factor for smoothing the tentative prolongation
- REAL ∗ amli_coef

  coefficients of the polynomial used by AMLI cycle
- AMG_data ∗ mgl_data

  AMG preconditioner data.
- ILU_data ∗ LU

  ILU preconditioner data (needed for CPR type preconditioner)
- dCSRmat ∗ A

  Matrix data.
- dCSRmat ∗ A_nk

  Matrix data for near kernel.
- dCSRmat ∗ P_nk

  Prolongation for near kernel.
- dCSRmat ∗ R_nk

  Restriction for near kernel.
- dvector r

  temporary dvector used to store and restore the residual
- REAL ∗ w

  temporary work space for other usage

9.41.1 Detailed Description

Data passed to the preconditioners.

Definition at line 842 of file fasp.h.

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

- fasp.h

9.42 precond_data_bsr Struct Reference

Data passed to the preconditioners.

```
#include <fasp_block.h>
```

Data Fields

- SHORT AMG_type

  type of AMG method
- SHORT print_level

  print level in AMG preconditioner
- INT maxit

  max number of iterations of AMG preconditioner
- INT max_levels

  max number of AMG levels
- REAL tol

  tolerance for AMG preconditioner
- SHORT cycle_type

  AMG cycle type.
- SHORT smoother

  AMG smoother type.
- SHORT smooth_order

  AMG smoother ordering.
- SHORT presmooth_iter

  number of presmoothing
- SHORT postsmooth_iter

  number of postsmoothing
- SHORT coarsening_type

  coarsening type
- REAL relaxation

  relaxation parameter for SOR smoother
- SHORT coarse_solver

  coarse solver type for AMG
- SHORT coarse_scaling

  switch of scaling of the coarse grid correction
- SHORT amli_degree

  degree of the polynomial used by AMLI cycle
- REAL ∗ amli_coef

  coefficients of the polynomial used by AMLI cycle
- REAL tentative_smooth

  smooth factor for smoothing the tentative prolongation
- SHORT nl_amli_krylov_type

  type of krylov method used by Nonlinear AMLI cycle
- AMG_data_bsr ∗ mgl_data

  AMG preconditioner data.
- AMG_data ∗ pres_mgl_data

  AMG preconditioner data for pressure block.
- ILU_data ∗ LU

  ILU preconditioner data (needed for CPR type preconditioner)
- dBSRmat ∗ A

  Matrix data.
- dCSRmat ∗ A_nk

Matrix data for near kernal.

- dCSRmat ∗ P_nk

  Prolongation for near kernal.

- dCSRmat ∗ R_nk

  Resriction for near kernal.

- dvector r

  temporary dvector used to store and restore the residual

- REAL ∗ w

  temporary work space for other usage

9.42.1 Detailed Description

Data passed to the preconditioners.

Note

This structure is needed for the AMG solver/preconditioner in BSR format

Definition at line 311 of file fasp_block.h.

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

- fasp_block.h

9.43 precond_data_str Struct Reference

Data passed to the preconditioner for dSTRmat matrices.

```
#include <fasp.h>
```

Data Fields

- SHORT AMG_type

  type of AMG method

- SHORT print_level

  print level in AMG preconditioner

- INT maxit

  max number of iterations of AMG preconditioner

- SHORT max_levels

  max number of AMG levels

- REAL tol

  tolerance for AMG preconditioner

- SHORT cycle_type

  AMG cycle type.

- SHORT smoother

  AMG smoother type.
- SHORT presmooth_iter

  number of presmoothing
- SHORT postsmooth_iter

  number of postsmoothing
- SHORT coarsening_type

  coarsening type
- REAL relaxation

  relaxation parameter for SOR smoother
- SHORT coarse_scaling

  switch of scaling of the coarse grid correction
- AMG_data ∗ mgl_data

  AMG preconditioner data.
- ILU_data ∗ LU

  ILU preconditioner data (needed for CPR type preconditioner)
- SHORT scaled

  whether the matrix are scaled or not
- dCSRmat ∗ A

  the original CSR matrix
- dSTRmat ∗ A_str

  store the whole reservoir block in STR format
- dSTRmat ∗ SS_str

  store Saturation block in STR format
- dvector ∗ diaginv

  the inverse of the diagonals for GS/block GS smoother (whole reservoir matrix)
- ivector ∗ pivot

  the pivot for the GS/block GS smoother (whole reservoir matrix)
- dvector ∗ diaginvS

  the inverse of the diagonals for GS/block GS smoother (saturation block)
- ivector ∗ pivotS

  the pivot for the GS/block GS smoother (saturation block)
- ivector ∗ order

  order for smoothing
- ivector ∗ neigh

  array to store neighbor information
- dvector r

  temporary dvector used to store and restore the residual
- REAL ∗ w

  temporary work space for other usage

9.43.1 Detailed Description

Data passed to the preconditioner for dSTRmat matrices.

Definition at line 938 of file fasp.h.

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

- fasp.h

9.44 precond_diagbsr Struct Reference

Data passed to diagnal preconditioner for dBSRmat matrices.

```
#include <fasp_block.h>
```

Data Fields

- INT nb

  dimension of each sub-block
- dvector diag

  diagnal elements

9.44.1 Detailed Description

Data passed to diagnal preconditioner for dBSRmat matrices.

Note

This is needed for the diagnal preconditioner.

Definition at line 293 of file fasp_block.h.

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

- fasp_block.h

9.45 precond_diagstr Struct Reference

Data passed to diagonal preconditioner for dSTRmat matrices.

```
#include <fasp.h>
```

Data Fields

- INT nc

  number of components
- dvector diag

  diagonal elements

9.45.1 Detailed Description

Data passed to diagonal preconditioner for dSTRmat matrices.

Note

This is needed for the diagonal preconditioner.

Definition at line 1030 of file fasp.h.

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

- fasp.h

9.46 precond_FASP_blkoil_data Struct Reference

Data passed to the preconditioner for preconditioning reservoir simulation problems.

```
#include <fasp_block.h>
```

Data Fields

- block_BSR ∗ A
  Part 1: Basic data.
- SHORT scaled
  Part 2: Data for CPR-like preconditioner for reservoir block.
- dvector ∗ diaginv_noscale
- dBSRmat ∗ RR
- ivector ∗ neigh
- ivector ∗ order
- dBSRmat ∗ SS
- dvector ∗ diaginv_S
- ivector ∗ pivot_S
- ILU_data ∗ LU_S
- dCSRmat ∗ PP
- AMG_data ∗ mgl_data
- ILU_data ∗ LU_P
- SHORT print_level
  print level in AMG preconditioner
- INT maxit_AMG
  max number of iterations of AMG preconditioner
- SHORT max_levels
  max number of AMG levels
- REAL amg_tol
  tolerance for AMG preconditioner
- SHORT cycle_type

AMG cycle type.
- SHORT smoother

  AMG smoother type.
- SHORT smooth_order

  AMG smoothing order.
- SHORT presmooth_iter

  number of presmoothing
- SHORT postsmooth_iter

  number of postsmoothing
- SHORT coarsening_type

  coarsening type
- INT coarse_dof

  coarset dof
- SHORT coarse_solver

  coarse level solver type
- REAL relaxation

  relaxation parameter for SOR smoother
- SHORT coarse_scaling

  switch of scaling of coarse grid correction
- SHORT amli_degree

  degree of the polynomial used by AMLI cycle
- REAL ∗ amli_coef

  coefficients of the polynomial used by AMLI cycle
- REAL tentative_smooth

  relaxation parameter for smoothing the tentative prolongation
- dvector ∗ diaginv
- ivector ∗ pivot
- ILU_data ∗ LU

  data of ILU for reservoir block
- ivector ∗ perf_idx
- ivector ∗ perf_neigh
- dCSRmat ∗ WW
- void ∗ Numeric

  data for direct solver for argumented well block
- REAL ∗ invS

  inverse of the schur complement (-I - Awr∗Arr^{-1}∗Arw)^{-1}, Arr may be replaced by LU
- INT maxit
- INT restart
- REAL tol
- dvector r
- REAL ∗ w

9.46.1 Detailed Description

Data passed to the preconditioner for preconditioning reservoir simulation problems.

Note

This is only needed for the Black Oil model with wells

Definition at line 532 of file fasp_block.h.

9.46.2 Field Documentation

9.46.2.1 A

```
block_BSR* A
```

Part 1: Basic data.

whole jacobian system in block_BSRmat

Definition at line 537 of file fasp_block.h.

9.46.2.2 diaginv

```
dvector* diaginv
```

inverse of the diagonal blocks of reservoir block

Definition at line 618 of file fasp_block.h.

9.46.2.3 diaginv_noscale

```
dvector* diaginv_noscale
```

inverse of diagonal blocks for diagonal scaling

Definition at line 544 of file fasp_block.h.

9.46.2.4 diaginv_S

```
dvector* diaginv_S
```

inverse of the diagonal blocks of saturation block

Definition at line 553 of file fasp_block.h.

9.46.2.5 LU_P

```
ILU_data* LU_P
```

ILU setup data for pressure block

Definition at line 564 of file fasp_block.h.

9.46.2.6 LU_S

```
ILU_data* LU_S
```

ILU setup data for saturation block

Definition at line 557 of file fasp_block.h.

9.46.2.7 maxit

```
INT maxit
```

max number of iterations

Definition at line 636 of file fasp_block.h.

9.46.2.8 mgl_data

```
AMG_data* mgl_data
```

AMG data for presure-presure block

Definition at line 561 of file fasp_block.h.

9.46.2.9 neigh

```
ivector* neigh
```

neighbor information of the reservoir block

Definition at line 548 of file fasp_block.h.

9.46.2.10 order

```
ivector* order
```

ordering of the reservoir block

Definition at line 549 of file fasp_block.h.

9.46.2.11 perf_idx

```
ivector* perf_idx
```

index of blocks which have perforation

Definition at line 625 of file fasp_block.h.

9.46.2.12 perf_neigh

`ivector* perf_neigh`

index of blocks which are neighbors of perforations (include perforations)

Definition at line 626 of file fasp_block.h.

9.46.2.13 pivot

`ivector* pivot`

pivot for the GS smoothers for the reservoir matrix

Definition at line 619 of file fasp_block.h.

9.46.2.14 pivot_S

`ivector* pivot_S`

pivoting for the GS smoothers for saturation block

Definition at line 554 of file fasp_block.h.

9.46.2.15 PP

`dCSRmat* PP`

pressure block

Definition at line 560 of file fasp_block.h.

9.46.2.16 r

`dvector r`

temporary dvector used to store and restore the residual

Definition at line 641 of file fasp_block.h.

9.46.2.17 restart

`INT restart`

number of iterations for restart

Definition at line 637 of file fasp_block.h.

9.46.2.18 RR

```
dBSRmat* RR
```

reservoir block

Definition at line 545 of file fasp_block.h.

9.46.2.19 scaled

```
SHORT scaled
```

Part 2: Data for CPR-like preconditioner for reservoir block.

scaled = 1 means the the following RR block is diagonal scaled

Definition at line 543 of file fasp_block.h.

9.46.2.20 SS

```
dBSRmat* SS
```

saturation block

Definition at line 552 of file fasp_block.h.

9.46.2.21 tol

```
REAL tol
```

tolerance

Definition at line 638 of file fasp_block.h.

9.46.2.22 w

```
REAL* w
```

temporary work space for other usage

Definition at line 642 of file fasp_block.h.

9.46.2.23 WW

```
dCSRmat* WW
```

Argumented well block

Definition at line 627 of file fasp_block.h.

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

- fasp_block.h

9.47 precond_sweeping_data Struct Reference

Data passed to the preconditioner for sweeping preconditioning.

```
#include <fasp_block.h>
```

Data Fields

- INT NumLayers
- dBLCmat * A
- dBLCmat * Ai
- dCSRmat * local_A
- void ** local_LU
- ivector * local_index
- dvector r
- REAL * w

9.47.1 Detailed Description

Data passed to the preconditioner for sweeping preconditioning.

Author

Xiaozhe Hu

Date

05/01/2014

Note

This is needed for the sweeping preconditioner.

Definition at line 655 of file fasp_block.h.

9.47.2 Field Documentation

9.47.2.1 A

`dBLCmat* A`

problem data, the sparse matrix

Definition at line 659 of file fasp_block.h.

9.47.2.2 Ai

`dBLCmat* Ai`

preconditioner data, the sparse matrix

Definition at line 660 of file fasp_block.h.

9.47.2.3 local_A

`dCSRmat* local_A`

local stiffness matrix for each layer

Definition at line 662 of file fasp_block.h.

9.47.2.4 local_index

`ivector* local_index`

local index for each layer

Definition at line 665 of file fasp_block.h.

9.47.2.5 local_LU

`void** local_LU`

lcoal LU decomposition (for UMFpack)

Definition at line 663 of file fasp_block.h.

9.47.2.6 NumLayers

```
INT NumLayers
```

number of layers

Definition at line 657 of file fasp_block.h.

9.47.2.7 r

```
dvector r
```

temporary dvector used to store and restore the residual

Definition at line 668 of file fasp_block.h.

9.47.2.8 w

```
REAL* w
```

temporary work space for other usage

Definition at line 669 of file fasp_block.h.

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

- fasp_block.h

9.48 Schwarz_data Struct Reference

Data for Schwarz methods.

```
#include <fasp.h>
```

Data Fields

- dCSRmat A
  pointer to the matrix
- INT nblk
  number of blocks
- INT * iblock
  row index of blocks
- INT * jblock
  column index of blocks
- REAL * rhsloc
  temp work space???
- dvector rhsloc1
  local right hand side
- dvector xloc1
  local solution
- REAL * au
  LU decomposition: the U block.
- REAL * al
  LU decomposition: the L block.
- INT Schwarz_type
  Schwarz method type.
- INT blk_solver
  Schwarz block solver.
- INT memt
  working space size
- INT * mask
  mask
- INT maxbs
  maximal block size
- INT * maxa
  maxa
- dCSRmat * blk_data
  matrix for each partition
- Mumps_data * mumps
  param for MUMPS
- Schwarz_param * swzparam
  param for Schwarz

9.48.1 Detailed Description

Data for Schwarz methods.

This is needed for the Schwarz solver/preconditioner/smoother.

Definition at line 540 of file fasp.h.

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

- fasp.h

9.49 Schwarz_param Struct Reference

Parameters for Schwarz method.

```
#include <fasp.h>
```

Data Fields

- SHORT print_level

  print leve
- SHORT Schwarz_type

  type for Schwarz method
- INT Schwarz_maxlvl

  maximal level for constructing the blocks
- INT Schwarz_mmsize

  maximal size of blocks
- INT Schwarz_blksolver

  type of Schwarz block solver

9.49.1 Detailed Description

Parameters for Schwarz method.

Added on 05/14/2012

Definition at line 469 of file fasp.h.

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

- fasp.h

Chapter 10

File Documentation

10.1 amg.c File Reference

AMG method as an iterative solver (main file)

```
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_solver_amg (dCSRmat ∗A, dvector ∗b, dvector ∗x, AMG_param ∗param)

  Solve Ax = b by algebraic multigrid methods.

10.1.1 Detailed Description

AMG method as an iterative solver (main file)

10.1.2 Function Documentation

10.1.2.1 fasp_solver_amg()

```
void fasp_solver_amg (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            AMG_param * param )
```

Solve Ax = b by algebraic multigrid methods.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns
param	Pointer to AMG_param: AMG parameters

Author

Chensong Zhang

Date

04/06/2010

Note

Refer to "Multigrid" by U. Trottenberg, C. W. Oosterlee and A. Schuller Appendix A.7 (by A. Brandt, P. Oswald and K. Stuben) Academic Press Inc., San Diego, CA, 2001.

Modified by Chensong Zhang on 01/10/2012 Modified by Chensong Zhang on 07/26/2014: Add error handling for AMG setup

Definition at line 37 of file amg.c.

10.2 amg_setup_cr.c File Reference

Brannick-Falgout compatible relaxation based AMG: SETUP phase.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- SHORT fasp_amg_setup_cr (AMG_data ∗mgl, AMG_param ∗param)

    Set up phase of Brannick Falgout CR coarsening for classic AMG.

10.2.1 Detailed Description

Brannick-Falgout compatible relaxation based AMG: SETUP phase.

Note

Setup A, P, R and levels using the Compatible Relaxation coarsening for classic AMG interpolation Refer to J. Brannick and R. Falgout "Compatible relaxation and coarsening in AMG"

Warning

Not working. Yet need to be fixed. –Chensong

10.2.2 Function Documentation

10.2.2.1 fasp_amg_setup_cr()

```
SHORT fasp_amg_setup_cr (
            AMG_data * mgl,
            AMG_param * param )
```

Set up phase of Brannick Falgout CR coarsening for classic AMG.

Parameters

mgl	Pointer to AMG data: AMG_data
param	Pointer to AMG parameters: AMG_param

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

James Brannick

Date

04/21/2010

Modified by Chensong Zhang on 05/10/2013: adjust the structure.

Definition at line 38 of file amg_setup_cr.c.

10.3 amg_setup_rs.c File Reference

Ruge-Stuben AMG: SETUP phase.

```
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- SHORT fasp_amg_setup_rs (AMG_data ∗mgl, AMG_param ∗param)
  Setup phase of Ruge and Stuben's classic AMG.

10.3.1 Detailed Description

Ruge-Stuben AMG: SETUP phase.

Note

Ref Multigrid by U. Trottenberg, C. W. Oosterlee and A. Schuller Appendix P475 A.7 (by A. Brandt, P. Oswald and K. Stuben) Academic Press Inc., San Diego, CA, 2001.

10.3.2 Function Documentation

10.3.2.1 fasp_amg_setup_rs()

```
SHORT fasp_amg_setup_rs (
            AMG_data * mgl,
            AMG_param * param )
```

Setup phase of Ruge and Stuben's classic AMG.

Parameters

mgl	Pointer to AMG data: AMG_data
param	Pointer to AMG parameters: AMG_param

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Chensong Zhang

Date

05/09/2010

Modified by Chensong Zhang on 04/04/2009. Modified by Chensong Zhang on 05/09/2010. Modified by Zhiyang Zhou on 11/17/2010. Modified by Xiaozhe Hu on 01/23/2011: add AMLI cycle. Modified by Chensong zhang on 09/09/2011↩ : add min dof. Modified by Xiaozhe Hu on 04/24/2013: aggressive coarsening. Modified by Chensong Zhang on 05/03/2013: add error handling in setup. Modified by Chensong Zhang on 05/10/2013: adjust the structure. Modified by Chensong Zhang on 07/26/2014: handle coarsening errors. Modified by Chensong Zhang on 09/23/2014: check coarse spaces.

Definition at line 47 of file amg_setup_rs.c.

10.4 amg_setup_sa.c File Reference

Smoothed aggregation AMG: SETUP phase.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "aggregation_csr.inl"
#include "aggregation_bsr.inl"
```

Functions

- SHORT fasp_amg_setup_sa (AMG_data ∗mgl, AMG_param ∗param)
  Set up phase of smoothed aggregation AMG.
- SHORT fasp_amg_setup_sa_bsr (AMG_data_bsr ∗mgl, AMG_param ∗param)
  Set up phase of smoothed aggregation AMG (BSR format)

10.4.1 Detailed Description

Smoothed aggregation AMG: SETUP phase.

Note

Setup A, P, PT and levels using the unsmoothed aggregation algorithm; Refer to P. Vanek, J. Madel and M. Brezina "Algebraic Multigrid on Unstructured Meshes", 1994

10.4.2 Function Documentation

10.4.2.1 fasp_amg_setup_sa()

```
SHORT fasp_amg_setup_sa (
            AMG_data * mgl,
            AMG_param * param )
```

Set up phase of smoothed aggregation AMG.

Parameters

mgl	Pointer to AMG data: AMG_data
param	Pointer to AMG parameters: AMG_param

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Xiaozhe Hu

Date

09/29/2009

Modified by Chensong Zhang on 04/06/2010. Modified by Chensong Zhang on 05/09/2010. Modified by Xiaozhe Hu on 01/23/2011: add AMLI cycle. Modified by Chensong Zhang on 05/10/2013: adjust the structure.

Definition at line 48 of file amg_setup_sa.c.

10.4.2.2 fasp_amg_setup_sa_bsr()

```
INT fasp_amg_setup_sa_bsr (
            AMG_data_bsr * mgl,
            AMG_param * param )
```

Set up phase of smoothed aggregation AMG (BSR format)

Parameters

mgl	Pointer to AMG data: AMG_data_bsr
param	Pointer to AMG parameters: AMG_param

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Xiaozhe Hu

Date

05/26/2014

Definition at line 85 of file amg_setup_sa.c.

10.5 amg_setup_ua.c File Reference

Unsmoothed aggregation AMG: SETUP phase.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "aggregation_csr.inl"
#include "aggregation_bsr.inl"
```

Functions

- SHORT fasp_amg_setup_ua (AMG_data ∗mgl, AMG_param ∗param)

  Set up phase of unsmoothed aggregation AMG.
- SHORT fasp_amg_setup_ua_bsr (AMG_data_bsr ∗mgl, AMG_param ∗param)

  Set up phase of unsmoothed aggregation AMG (BSR format)

10.5.1 Detailed Description

Unsmoothed aggregation AMG: SETUP phase.

Note

Setup A, P, PT and levels using the unsmoothed aggregation algorithm; Refer to P. Vanek, J. Madel and M. Brezina "Algebraic Multigrid on Unstructured Meshes", 1994

10.5.2 Function Documentation

10.5.2.1 fasp_amg_setup_ua()

```
SHORT fasp_amg_setup_ua (
            AMG_data * mgl,
            AMG_param * param )
```

Set up phase of unsmoothed aggregation AMG.

Parameters

mgl	Pointer to AMG data: AMG_data
param	Pointer to AMG parameters: AMG_param

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Xiaozhe Hu

Date

12/28/2011

Definition at line 38 of file amg_setup_ua.c.

10.5.2.2 fasp_amg_setup_ua_bsr()

```
INT fasp_amg_setup_ua_bsr (
            AMG_data_bsr * mgl,
            AMG_param * param )
```

Set up phase of unsmoothed aggregation AMG (BSR format)

Parameters

mgl	Pointer to AMG data: AMG_data_bsr
param	Pointer to AMG parameters: AMG_param

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Xiaozhe Hu

Date

03/16/2012

Definition at line 69 of file amg_setup_ua.c.

10.6 amg_solve.c File Reference

Algebraic multigrid iterations: SOLVE phase.

```
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_amg_solve (AMG_data *mgl, AMG_param *param)

  AMG – SOLVE phase.
- INT fasp_amg_solve_amli (AMG_data *mgl, AMG_param *param)

  AMLI – SOLVE phase.
- INT fasp_amg_solve_nl_amli (AMG_data *mgl, AMG_param *param)

  Nonlinear AMLI – SOLVE phase.
- void fasp_famg_solve (AMG_data *mgl, AMG_param *param)

  FMG – SOLVE phase.

10.6.1 Detailed Description

Algebraic multigrid iterations: SOLVE phase.

Note

Solve Ax=b using multigrid method. This is SOLVE phase only and is independent of SETUP method used! Should be called after multigrid hierarchy has been generated!

10.6.2 Function Documentation

10.6.2.1 fasp_amg_solve()

```
INT fasp_amg_solve (
            AMG_data * mgl,
            AMG_param * param )
```

AMG – SOLVE phase.

Parameters

mgl	Pointer to AMG data: AMG_data
param	Pointer to AMG parameters: AMG_param

Returns

Iteration number if converges; ERROR otherwise.

Author

Xuehai Huang, Chensong Zhang

Date

04/02/2010

Modified by Chensong 04/21/2013: Fix an output typo

Definition at line 36 of file amg_solve.c.

10.6.2.2 fasp_amg_solve_amli()

```
INT fasp_amg_solve_amli (
            AMG_data * mgl,
            AMG_param * param )
```

AMLI – SOLVE phase.

Parameters

mgl	Pointer to AMG data: AMG_data
param	Pointer to AMG parameters: AMG_param

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

01/23/2011

Note

AMLI polynomial computed by the best approximation of 1/x. Refer to Johannes K. Kraus, Panayot S. Vassilevski, Ludmil T. Zikatanov, "Polynomial of best uniform approximation to x^{-1} and smoothing in two-level methods", 2013.

Modified by Chensong 04/21/2013: Fix an output typo

Definition at line 125 of file amg_solve.c.

10.6.2.3 fasp_amg_solve_nl_amli()

```
INT fasp_amg_solve_nl_amli (
            AMG_data * mgl,
            AMG_param * param )
```

Nonlinear AMLI – SOLVE phase.

Parameters

mgl	Pointer to AMG data: AMG_data
param	Pointer to AMG parameters: AMG_param

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

04/30/2011

Modified by Chensong 04/21/2013: Fix an output typo

Note

Nonlinear AMLI-cycle. Refer to Xiazhe Hu, Panayot S. Vassilevski, Jinchao Xu "Comparative Convergence Analysis of Nonlinear AMLI-cycle Multigrid", 2013.

Definition at line 209 of file amg_solve.c.

10.6.2.4 fasp_famg_solve()

```
void fasp_famg_solve (
            AMG_data * mgl,
            AMG_param * param )
```

FMG – SOLVE phase.

Parameters

mgl	Pointer to AMG data: AMG_data
param	Pointer to AMG parameters: AMG_param

Author

Chensong Zhang

Date

01/10/2012

Definition at line 281 of file amg_solve.c.

10.7 amlirecur.c File Reference

Abstract AMLI multilevel iteration – recursive version.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "mg_util.inl"
```

Functions

- void fasp_solver_amli (AMG_data ∗mgl, AMG_param ∗param, INT level)
  Solve Ax=b with recursive AMLI-cycle.
- void fasp_solver_nl_amli (AMG_data ∗mgl, AMG_param ∗param, INT level, INT num_levels)
  Solve Ax=b with recursive nonlinear AMLI-cycle.
- void fasp_solver_nl_amli_bsr (AMG_data_bsr ∗mgl, AMG_param ∗param, INT level, INT num_levels)
  Solve Ax=b with recursive nonlinear AMLI-cycle.
- void fasp_amg_amli_coef (const REAL lambda_max, const REAL lambda_min, const INT degree, REAL ∗coef)
  Compute the coefficients of the polynomial used by AMLI-cycle.

10.7.1 Detailed Description

Abstract AMLI multilevel iteration – recursive version.

Note

AMLI and non-linear AMLI cycles

10.7.2 Function Documentation

10.7.2.1 fasp_amg_amli_coef()

```
void fasp_amg_amli_coef (
            const REAL lambda_max,
            const REAL lambda_min,
            const INT degree,
            REAL * coef )
```

Compute the coefficients of the polynomial used by AMLI-cycle.

Parameters

lambda_max	Maximal lambda
lambda_min	Minimal lambda
degree	Degree of polynomial approximation
coef	Coefficient of AMLI (output)

Author

Xiaozhe Hu

Date

01/23/2011

Definition at line 706 of file amlirecur.c.

10.7.2.2 fasp_solver_amli()

```
void fasp_solver_amli (
            AMG_data * mgl,
            AMG_param * param,
            INT level )
```

Solve Ax=b with recursive AMLI-cycle.

Parameters

mgl	Pointer to AMG data: AMG_data
param	Pointer to AMG parameters: AMG_param
level	Current level

Author

Xiaozhe Hu

Date

01/23/2011

Note

AMLI polynomial computed by the best approximation of 1/x. Refer to Johannes K. Kraus, Panayot S. Vassilevski, Ludmil T. Zikatanov, "Polynomial of best uniform approximation to x^{-1} and smoothing in two-level methods", 2013.

Modified by Chensong Zhang on 02/27/2013: update direct solvers. Modified by Zheng Li on 11/10/2014: update direct solvers. Modified by Hongxuan Zhang on 12/15/2015: update direct solvers.

Definition at line 45 of file amlirecur.c.

10.7.2.3 fasp_solver_nl_amli()

```
void fasp_solver_nl_amli (
            AMG_data * mgl,
            AMG_param * param,
            INT level,
            INT num_levels )
```

Solve Ax=b with recursive nonlinear AMLI-cycle.

Parameters

mgl	Pointer to AMG_data data
param	Pointer to AMG parameters
level	Current level
num_levels	Total number of levels

Author

Xiaozhe Hu

Date

04/06/2010

Note

Refer to Xiazhe Hu, Panayot S. Vassilevski, Jinchao Xu "Comparative Convergence Analysis of Nonlinear AMLI-cycle Multigrid", 2013.

Modified by Chensong Zhang on 02/27/2013: update direct solvers. Modified by Zheng Li on 11/10/2014: update direct solvers. Modified by Hongxuan Zhang on 12/15/2015: update direct solvers.

Definition at line 269 of file amlirecur.c.

10.7.2.4 fasp_solver_nl_amli_bsr()

```
void fasp_solver_nl_amli_bsr (
            AMG_data_bsr * mgl,
            AMG_param * param,
            INT level,
            INT num_levels )
```

Solve Ax=b with recursive nonlinear AMLI-cycle.

Parameters

mgl	Pointer to AMG data: AMG_data
param	Pointer to AMG parameters: AMG_param
level	Current level
num_levels	Total number of levels

Author

Xiaozhe Hu

Date

04/06/2010

Note

Nonlinear AMLI-cycle. Refer to Xiazhe Hu, Panayot S. Vassilevski, Jinchao Xu "Comparative Convergence Analysis of Nonlinear AMLI-cycle Multigrid", 2013.

Modified by Chensong Zhang on 02/27/2013: update direct solvers. Modified by Hongxuan Zhang on 12/15/2015: update direct solvers.

Definition at line 508 of file amlirecur.c.

10.8 array.c File Reference

Simple array operations – init, set, copy, etc.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_array_null (REAL ∗x)
  Initialize an array.
- void fasp_array_set (const INT n, REAL ∗x, const REAL val)
  Set initial value for an array to be x=val.
- void fasp_iarray_set (const INT n, INT ∗x, const INT val)
  Set initial value for an array to be x=val.
- void fasp_array_cp (const INT n, REAL ∗x, REAL ∗y)
  Copy an array to the other y=x.
- void fasp_iarray_cp (const INT n, INT ∗x, INT ∗y)
  Copy an array to the other y=x.
- void fasp_array_cp_nc3 (REAL ∗x, REAL ∗y)
  Copy an array to the other y=x, the length is 3.
- void fasp_array_cp_nc5 (REAL ∗x, REAL ∗y)
  Copy an array to the other y=x, the length is 5.
- void fasp_array_cp_nc7 (REAL ∗x, REAL ∗y)
  Copy an array to the other y=x, the length is 7.
- void fasp_array_permut_nb (INT n, INT nb, REAL ∗x, INT ∗p, REAL ∗y)
  Array mapping.
- void fasp_array_invpermut_nb (INT n, INT nb, REAL ∗x, INT ∗p, REAL ∗y)
  Array mapping.

10.8.1 Detailed Description

Simple array operations – init, set, copy, etc.

10.8.2 Function Documentation

10.8.2.1 fasp_array_cp()

```
void fasp_array_cp (
            const INT n,
            REAL * x,
            REAL * y )
```

Copy an array to the other y=x.

Parameters

n	Number of variables
x	Pointer to the original vector
y	Pointer to the destination vector

Author

Chensong Zhang

Date

2010/04/03

Definition at line 165 of file array.c.

10.8.2.2 fasp_array_cp_nc3()

```
void fasp_array_cp_nc3 (
            REAL * x,
            REAL * y )
```

Copy an array to the other y=x, the length is 3.

Parameters

x	Pointer to the original vector
y	Pointer to the destination vector

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Note

Special unrolled routine designed for a specific application

Definition at line 205 of file array.c.

10.8.2.3 fasp_array_cp_nc5()

```
void fasp_array_cp_nc5 (
            REAL * x,
            REAL * y )
```

Copy an array to the other y=x, the length is 5.

Parameters

x	Pointer to the original vector
y	Pointer to the destination vector

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Note

Special unrolled routine designed for a specific application

Definition at line 226 of file array.c.

10.8.2.4 fasp_array_cp_nc7()

```
void fasp_array_cp_nc7 (
            REAL * x,
            REAL * y )
```

Copy an array to the other y=x, the length is 7.

Parameters

x	Pointer to the original vector
y	Pointer to the destination vector

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Note

Special unrolled routine designed for a specific application

Definition at line 249 of file array.c.

10.8.2.5 fasp_array_invpermut_nb()

```
void fasp_array_invpermut_nb (
            INT n,
            INT nb,
            REAL * x,
            INT * p,
            REAL * y )
```

Array mapping.

Parameters

n	Size of array
nb	Step size
x	Pointer to the original vector
p	Pointer to index mapping
y	Pointer to the destination vector

Author

Zheng Li

Date

12/04/2016

Definition at line 312 of file array.c.

10.8.2.6 fasp_array_null()

```
void fasp_array_null (
            REAL * x )
```

Initialize an array.

Parameters

x	Pointer to the vector

Author

Chensong Zhang

Date

2010/04/03

Definition at line 29 of file array.c.

10.8.2.7 fasp_array_permut_nb()

```
void fasp_array_permut_nb (
            INT n,
            INT nb,
            REAL * x,
            INT * p,
            REAL * y )
```

Array mapping.

Parameters

n	Size of array
nb	Step size
x	Pointer to the original vector
p	Pointer to index mapping
y	Pointer to the destination vector

Author

Zheng Li

Date

12/04/2016

Definition at line 276 of file array.c.

10.8.2.8 fasp_array_set()

```
void fasp_array_set (
            const INT n,
            REAL * x,
            const REAL val )
```

Set initial value for an array to be x=val.

Parameters

n	Number of variables
x	Pointer to the vector
val	Initial value for the REAL array

Author

Chensong Zhang

Date

04/03/2010

Modified by Chunsheng Feng, Xiaoqiang Yue on 05/23/2012

Definition at line 48 of file array.c.

10.8.2.9 fasp_iarray_cp()

```
void fasp_iarray_cp (
            const INT n,
            INT * x,
            INT * y )
```

Copy an array to the other y=x.

Parameters

n	Number of variables
x	Pointer to the original vector
y	Pointer to the destination vector

Author

Chunsheng Feng, Xiaoqiang Yue

Date

05/23/2012

Definition at line 185 of file array.c.

10.8.2.10 fasp_iarray_set()

```
void fasp_iarray_set (
            const INT n,
            INT * x,
            const INT val )
```

Set initial value for an array to be x=val.

Parameters

n	Number of variables
x	Pointer to the vector
val	Initial value for the REAL array

Author

Chensong Zhang

Date

04/03/2010

Modified by Chunsheng Feng, Xiaoqiang Yue on 05/25/2012

Definition at line 107 of file array.c.

10.9 blas_array.c File Reference

BLAS1 operations for arrays.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_blas_array_ax (const INT n, const REAL a, REAL ∗x)

    x = a∗x
- void fasp_blas_array_axpy (const INT n, const REAL a, REAL ∗x, REAL ∗y)

    y = a∗x + y
- void fasp_blas_array_axpyz (const INT n, const REAL a, REAL ∗x, REAL ∗y, REAL ∗z)

    z = a∗x + y
- void fasp_blas_array_axpby (const INT n, const REAL a, REAL ∗x, const REAL b, REAL ∗y)

    y = a∗x + b∗y
- REAL fasp_blas_array_dotprod (const INT n, const REAL ∗x, const REAL ∗y)

    Inner product of two arraies (x,y)
- REAL fasp_blas_array_norm1 (const INT n, const REAL ∗x)

    L1 norm of array x.
- REAL fasp_blas_array_norm2 (const INT n, const REAL ∗x)

    L2 norm of array x.
- REAL fasp_blas_array_norminf (const INT n, const REAL ∗x)

    Linf norm of array x.

10.9.1 Detailed Description

BLAS1 operations for arrays.

10.9.2 Function Documentation

10.9.2.1 fasp_blas_array_ax()

```
void fasp_blas_array_ax (
            const INT n,
            const REAL a,
            REAL * x )
```

x = a∗x

Parameters

n	Number of variables
a	Factor a
x	Pointer to x

Author

Chensong Zhang

Date

07/01/209

Modified by Chunsheng Feng, Xiaoqiang Yue on 05/23/2012

Note

x is reused to store the resulting array.

Definition at line 35 of file blas_array.c.

10.9.2.2 fasp_blas_array_axpby()

```
void fasp_blas_array_axpby (
            const INT n,
            const REAL a,
            REAL * x,
            const REAL b,
            REAL * y )
```

y = a∗x + b∗y

Parameters

n	Number of variables
a	Factor a
x	Pointer to x
b	Factor b
y	Pointer to y

Author

Chensong Zhang

Date

07/01/209

Modified by Chunsheng Feng, Xiaoqiang Yue on 05/23/2012

Note

y is reused to store the resulting array.

Definition at line 218 of file blas_array.c.

10.9.2.3 fasp_blas_array_axpy()

```
void fasp_blas_array_axpy (
            const INT n,
            const REAL a,
            REAL * x,
            REAL * y )
```

y = a∗x + y

Parameters

n	Number of variables
a	Factor a
x	Pointer to x
y	Pointer to y

Author

Chensong Zhang

Date

07/01/209

Modified by Chunsheng Feng, Xiaoqiang Yue on 05/23/2012

Note

y is reused to store the resulting array.

Definition at line 87 of file blas_array.c.

10.9.2.4 fasp_blas_array_axpyz()

```
void fasp_blas_array_axpyz (
            const INT n,
            const REAL a,
            REAL * x,
            REAL * y,
            REAL * z )
```

z = a∗x + y

Parameters

n	Number of variables
a	Factor a
x	Pointer to x
y	Pointer to y
z	Pointer to z

Author

Chensong Zhang

Date

07/01/209

Modified by Chunsheng Feng, Xiaoqiang Yue on 05/23/2012

Definition at line 167 of file blas_array.c.

10.9.2.5 fasp_blas_array_dotprod()

```
REAL fasp_blas_array_dotprod (
            const INT n,
            const REAL * x,
            const REAL * y )
```

Inner product of two arraies (x,y)

Parameters

n	Number of variables
x	Pointer to x
y	Pointer to y

Returns

Inner product (x,y)

Author

Chensong Zhang

Date

07/01/209

Modified by Chunsheng Feng, Xiaoqiang Yue on 05/23/2012

Definition at line 267 of file blas_array.c.

10.9.2.6 fasp_blas_array_norm1()

```
REAL fasp_blas_array_norm1 (
            const INT n,
            const REAL * x )
```

L1 norm of array x.

Parameters

n	Number of variables
x	Pointer to x

Returns

L1 norm of x

Author

Chensong Zhang

Date

07/01/209

Modified by Chunsheng Feng, Xiaoqiang Yue on 05/23/2012

Definition at line 307 of file blas_array.c.

10.9.2.7 fasp_blas_array_norm2()

```
REAL fasp_blas_array_norm2 (
            const INT n,
            const REAL * x )
```

L2 norm of array x.

Parameters

n	Number of variables
x	Pointer to x

Returns

L2 norm of x

Author

Chensong Zhang

Date

07/01/209

Modified by Chunsheng Feng, Xiaoqiang Yue on 05/23/2012

Definition at line 347 of file blas_array.c.

10.9.2.8 fasp_blas_array_norminf()

```
REAL fasp_blas_array_norminf (
            const INT n,
            const REAL * x )
```

Linf norm of array x.

Parameters

n	Number of variables
x	Pointer to x

Returns

L_inf norm of x

Author

Chensong Zhang

Date

07/01/209

Modified by Chunsheng Feng, Zheng Li on 06/28/2012

Definition at line 388 of file blas_array.c.

10.10 blas_blc.c File Reference

BLAS2 operations for dBLCmat matrices.

```
#include <time.h>
#include "fasp.h"
#include "fasp_block.h"
#include "fasp_functs.h"
```

Functions

- void fasp_blas_dblc_aAxpy (const REAL alpha, dBLCmat ∗A, REAL ∗x, REAL ∗y)

  Matrix-vector multiplication y = alpha∗A∗x + y.
- void fasp_blas_dblc_mxv (dBLCmat ∗A, REAL ∗x, REAL ∗y)

  Matrix-vector multiplication y = A∗x.
- void fasp_blas_bdbsr_aAxpy (const REAL alpha, block_BSR ∗A, REAL ∗x, REAL ∗y)

  Matrix-vector multiplication y = alpha∗A∗x + y.
- void fasp_blas_bdbsr_mxv (block_BSR ∗A, REAL ∗x, REAL ∗y)

  Matrix-vector multiplication y = A∗x.

10.10.1 Detailed Description

BLAS2 operations for dBLCmat matrices.

10.10.2 Function Documentation

10.10.2.1 fasp_blas_bdbsr_aAxpy()

```
void fasp_blas_bdbsr_aAxpy (
            const REAL alpha,
            block_BSR * A,
            REAL * x,
            REAL * y )
```

Matrix-vector multiplication y = alpha∗A∗x + y.

Parameters

alpha	REAL factor a
A	Pointer to block_BSR matrix A
x	Pointer to array x
y	Pointer to array y

Author

Xiaozhe Hu

Date

11/11/2010

Definition at line 288 of file blas_blc.c.

10.10.2.2 fasp_blas_bdbsr_mxv()

```
void fasp_blas_bdbsr_mxv (
            block_BSR * A,
            REAL * x,
            REAL * y )
```

Matrix-vector multiplication y = A∗x.

Parameters

A	Pointer to block_BSR matrix A
x	Pointer to array x
y	Pointer to array y

Author

Xiaozhe Hu

Date

11/11/2010

Definition at line 326 of file blas_blc.c.

10.10.2.3 fasp_blas_dblc_aAxpy()

```
void fasp_blas_dblc_aAxpy (
            const REAL alpha,
            dBLCmat * A,
            REAL * x,
            REAL * y )
```

Matrix-vector multiplication y = alpha∗A∗x + y.

Parameters

alpha	REAL factor a
A	Pointer to dBLCmat matrix A
x	Pointer to array x
y	Pointer to array y

Author

Xiaozhe Hu

Date

06/04/2010

Definition at line 30 of file blas_blc.c.

10.10.2.4 fasp_blas_dblc_mxv()

```
void fasp_blas_dblc_mxv (
            dBLCmat * A,
            REAL * x,
            REAL * y )
```

Matrix-vector multiplication y = A∗x.

Parameters

A	Pointer to dBLCmat matrix A
x	Pointer to array x
y	Pointer to array y

Author

Chensong Zhang

Date

04/27/2013

Definition at line 155 of file blas_blc.c.

10.11 blas_bsr.c File Reference

BLAS2 operations for dBSRmat matrices.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_blas_dbsr_axm (dBSRmat ∗A, const REAL alpha)

  Multiply a sparse matrix A in BSR format by a scalar alpha.
- void fasp_blas_dbsr_aAxpby (const REAL alpha, dBSRmat ∗A, REAL ∗x, const REAL beta, REAL ∗y)

  Compute y := alpha∗A∗x + beta∗y.
- void fasp_blas_dbsr_aAxpy (const REAL alpha, dBSRmat ∗A, REAL ∗x, REAL ∗y)

  Compute y := alpha∗A∗x + y.
- void fasp_blas_dbsr_aAxpy_agg (const REAL alpha, dBSRmat ∗A, REAL ∗x, REAL ∗y)

  Compute y := alpha∗A∗x + y where each small block matrix is an identity matrix.
- void fasp_blas_dbsr_mxv (dBSRmat ∗A, REAL ∗x, REAL ∗y)

  Compute y := A∗x.
- void fasp_blas_dbsr_mxv_agg (dBSRmat ∗A, REAL ∗x, REAL ∗y)

  Compute y := A∗x, where each small block matrices of A is an identity matrix.
- void fasp_blas_dbsr_mxm (dBSRmat ∗A, dBSRmat ∗B, dBSRmat ∗C)

  Sparse matrix multiplication C=A∗B.
- void fasp_blas_dbsr_rap1 (dBSRmat ∗R, dBSRmat ∗A, dBSRmat ∗P, dBSRmat ∗B)

  dBSRmat sparse matrix multiplication B=R∗A∗P
- void fasp_blas_dbsr_rap (dBSRmat ∗R, dBSRmat ∗A, dBSRmat ∗P, dBSRmat ∗B)

  dBSRmat sparse matrix multiplication B=R∗A∗P
- void fasp_blas_dbsr_rap_agg (dBSRmat ∗R, dBSRmat ∗A, dBSRmat ∗P, dBSRmat ∗B)

  dBSRmat sparse matrix multiplication B=R∗A∗P, where small block matrices in P and R are identity matrices!

10.11.1 Detailed Description

BLAS2 operations for dBSRmat matrices.

10.11.2 Function Documentation

10.11.2.1 fasp_blas_dbsr_aAxpby()

```
void fasp_blas_dbsr_aAxpby (
            const REAL alpha,
            dBSRmat * A,
            REAL * x,
            const REAL beta,
            REAL * y )
```

Compute y := alpha∗A∗x + beta∗y.

Parameters

alpha	REAL factor alpha
A	Pointer to the dBSRmat matrix
x	Pointer to the array x
beta	REAL factor beta
y	Pointer to the array y

Author

Zhiyang Zhou

Date

10/25/2010

Modified by Chunsheng Feng, Zheng Li on 06/29/2012

Note

Works for general nb (Xiaozhe)

Definition at line 59 of file blas_bsr.c.

10.11.2.2 fasp_blas_dbsr_aAxpy()

```
void fasp_blas_dbsr_aAxpy (
            const REAL alpha,
            dBSRmat * A,
            REAL * x,
            REAL * y )
```

Compute y := alpha∗A∗x + y.

Parameters

alpha	REAL factor alpha
A	Pointer to the dBSRmat matrix
x	Pointer to the array x
y	Pointer to the array y

Author

Zhiyang Zhou

Date

10/25/2010

Modified by Chunsheng Feng, Xiaoqiang Yue on 05/23/2012

Note

Works for general nb (Xiaozhe)

Definition at line 339 of file blas_bsr.c.

10.11.2.3 fasp_blas_dbsr_aAxpy_agg()

```
void fasp_blas_dbsr_aAxpy_agg (
            const REAL alpha,
            dBSRmat * A,
            REAL * x,
            REAL * y )
```

Compute y := alpha∗A∗x + y where each small block matrix is an identity matrix.

Parameters

alpha	REAL factor alpha
A	Pointer to the dBSRmat matrix
x	Pointer to the array x
y	Pointer to the array y

Author

Xiaozhe Hu

Date

01/02/2014

Note

Works for general nb (Xiaozhe)

Definition at line 613 of file blas_bsr.c.

10.11.2.4 fasp_blas_dbsr_axm()

```
void fasp_blas_dbsr_axm (
            dBSRmat * A,
            const REAL alpha )
```

Multiply a sparse matrix A in BSR format by a scalar alpha.

Parameters

A	Pointer to dBSRmat matrix A
alpha	REAL factor alpha

Author

Xiaozhe Hu

Date

05/26/2014

Definition at line 30 of file blas_bsr.c.

10.11.2.5 fasp_blas_dbsr_mxm()

```
void fasp_blas_dbsr_mxm (
            dBSRmat * A,
            dBSRmat * B,
            dBSRmat * C )
```

Sparse matrix multiplication C=A∗B.

Parameters

A	Pointer to the dBSRmat matrix A
B	Pointer to the dBSRmat matrix B
C	Pointer to dBSRmat matrix equal to A∗B

Author

Xiaozhe Hu

Date

05/26/2014

Note

This fct will be replaced! – Xiaozhe

Definition at line 4634 of file blas_bsr.c.

10.11.2.6 fasp_blas_dbsr_mxv()

```
void fasp_blas_dbsr_mxv (
            dBSRmat * A,
            REAL * x,
            REAL * y )
```

Compute y := A∗x.

Parameters

A	Pointer to the dBSRmat matrix
x	Pointer to the array x
y	Pointer to the array y

Author

Zhiyang Zhou

Date

10/25/2010

Note

Works for general nb (Xiaozhe)

Modified by Chunsheng Feng, Xiaoqiang Yue on 05/23/2012

Definition at line 898 of file blas_bsr.c.

10.11.2.7 fasp_blas_dbsr_mxv_agg()

```
void fasp_blas_dbsr_mxv_agg (
            dBSRmat * A,
            REAL * x,
            REAL * y )
```

Compute y := A*x, where each small block matrices of A is an identity matrix.

Parameters

A	Pointer to the dBSRmat matrix
x	Pointer to the array x
y	Pointer to the array y

Author

Xiaozhe Hu

Date

01/02/2014

Note

Works for general nb (Xiaozhe)

Definition at line 2684 of file blas_bsr.c.

10.11.2.8 fasp_blas_dbsr_rap()

```
void fasp_blas_dbsr_rap (
            dBSRmat * R,
            dBSRmat * A,
            dBSRmat * P,
            dBSRmat * B )
```

dBSRmat sparse matrix multiplication B=R*A*P

Parameters

R	Pointer to the dBSRmat matrix
A	Pointer to the dBSRmat matrix
P	Pointer to the dBSRmat matrix
B	Pointer to dBSRmat matrix equal to R*A*P (output)

Author

Xiaozhe Hu, Chunsheng Feng, Zheng Li

Date

10/24/2012

Note

Ref. R.E. Bank and C.C. Douglas. SMMP: Sparse Matrix Multiplication Package. Advances in Computational Mathematics, 1 (1993), pp. 127-137.

Definition at line 4938 of file blas_bsr.c.

10.11.2.9 fasp_blas_dbsr_rap1()

```
void fasp_blas_dbsr_rap1 (
            dBSRmat * R,
            dBSRmat * A,
            dBSRmat * P,
            dBSRmat * B )
```

dBSRmat sparse matrix multiplication B=R∗A∗P

Parameters

R	Pointer to the dBSRmat matrix
A	Pointer to the dBSRmat matrix
P	Pointer to the dBSRmat matrix
B	Pointer to dBSRmat matrix equal to R∗A∗P (output)

Author

Chunsheng Feng, Xiaoqiang Yue and Xiaozhe Hu

Date

08/08/2011

Note

Ref. R.E. Bank and C.C. Douglas. SMMP: Sparse Matrix Multiplication Package. Advances in Computational Mathematics, 1 (1993), pp. 127-137.

Definition at line 4754 of file blas_bsr.c.

10.11.2.10 fasp_blas_dbsr_rap_agg()

```
void fasp_blas_dbsr_rap_agg (
            dBSRmat * R,
            dBSRmat * A,
            dBSRmat * P,
            dBSRmat * B )
```

dBSRmat sparse matrix multiplication B=R∗A∗P, where small block matrices in P and R are identity matrices!

Parameters

R	Pointer to the dBSRmat matrix
A	Pointer to the dBSRmat matrix
P	Pointer to the dBSRmat matrix
B	Pointer to dBSRmat matrix equal to R∗A∗P (output)

Author

Xiaozhe Hu

Date

10/24/2012

Note

Ref. R.E. Bank and C.C. Douglas. SMMP: Sparse Matrix Multiplication Package. Advances in Computational Mathematics, 1 (1993), pp. 127-137.

Definition at line 5203 of file blas_bsr.c.

10.12 blas_csr.c File Reference

BLAS2 operations for dCSRmat matrices.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- INT fasp_blas_dcsr_add (dCSRmat ∗A, const REAL alpha, dCSRmat ∗B, const REAL beta, dCSRmat ∗C)

  compute C = alpha∗A + beta∗B in CSR format
- void fasp_blas_dcsr_axm (dCSRmat ∗A, const REAL alpha)

  Multiply a sparse matrix A in CSR format by a scalar alpha.
- void fasp_blas_dcsr_mxv (dCSRmat ∗A, REAL ∗x, REAL ∗y)

  Matrix-vector multiplication y = A∗x.
- void fasp_blas_dcsr_mxv_agg (dCSRmat ∗A, REAL ∗x, REAL ∗y)

  Matrix-vector multiplication y = A∗x, where the entries of A are all ones.
- void fasp_blas_dcsr_aAxpy (const REAL alpha, dCSRmat ∗A, REAL ∗x, REAL ∗y)

  Matrix-vector multiplication y = alpha∗A∗x + y.
- void fasp_blas_dcsr_aAxpy_agg (const REAL alpha, dCSRmat ∗A, REAL ∗x, REAL ∗y)

  Matrix-vector multiplication y = alpha∗A∗x + y (the entries of A are all ones)
- REAL fasp_blas_dcsr_vmv (dCSRmat ∗A, REAL ∗x, REAL ∗y)

  vector-Matrix-vector multiplication alpha = y'∗A∗x
- void fasp_blas_dcsr_mxm (dCSRmat ∗A, dCSRmat ∗B, dCSRmat ∗C)

  Sparse matrix multiplication C=A∗B.
- void fasp_blas_dcsr_rap (dCSRmat ∗R, dCSRmat ∗A, dCSRmat ∗P, dCSRmat ∗RAP)

  Triple sparse matrix multiplication B=R∗A∗P.
- void fasp_blas_dcsr_rap_agg (dCSRmat ∗R, dCSRmat ∗A, dCSRmat ∗P, dCSRmat ∗RAP)

  Triple sparse matrix multiplication B=R∗A∗P.
- void fasp_blas_dcsr_rap_agg1 (dCSRmat ∗R, dCSRmat ∗A, dCSRmat ∗P, dCSRmat ∗B)

  Triple sparse matrix multiplication B=R∗A∗P (nonzero entries of R and P are ones)
- void fasp_blas_dcsr_ptap (dCSRmat ∗Pt, dCSRmat ∗A, dCSRmat ∗P, dCSRmat ∗Ac)

  Triple sparse matrix multiplication B=P'∗A∗P.
- void fasp_blas_dcsr_rap4 (dCSRmat ∗R, dCSRmat ∗A, dCSRmat ∗P, dCSRmat ∗B, INT ∗icor_ysk)

  Triple sparse matrix multiplication B=R∗A∗P.
- void fasp_blas_dcsr_bandwith (dCSRmat ∗A, INT ∗bndwith)

  Get bandwith of matrix.

10.12.1 Detailed Description

BLAS2 operations for dCSRmat matrices.

Note

Sparse functions usually contain three runs. The three runs are all the same but thy serve different purpose.

Example: If you do c=a+b:

- first do a dry run to find the number of non-zeroes in the result and form ic;
- allocate space (memory) for jc and form this one;
- if you only care about a "boolean" result of the addition, you stop here;
- you call another routine, which uses ic and jc to perform the addition.

10.12.2 Function Documentation

10.12.2.1 fasp_blas_dcsr_aAxpy()

```
void fasp_blas_dcsr_aAxpy (
            const REAL alpha,
            dCSRmat * A,
            REAL * x,
            REAL * y )
```

Matrix-vector multiplication y = alpha∗A∗x + y.

Parameters

alpha	REAL factor alpha
A	Pointer to dCSRmat matrix A
x	Pointer to array x
y	Pointer to array y

Author

Chensong Zhang

Date

07/01/2009

Modified by Chunsheng Feng, Xiaoqiang Yue on 05/26/2012

Definition at line 479 of file blas_csr.c.

10.12.2.2 fasp_blas_dcsr_aAxpy_agg()

```
void fasp_blas_dcsr_aAxpy_agg (
            const REAL alpha,
            dCSRmat * A,
            REAL * x,
            REAL * y )
```

Matrix-vector multiplication y = alpha∗A∗x + y (the entries of A are all ones)

Parameters

alpha	REAL factor alpha
A	Pointer to dCSRmat matrix A
x	Pointer to array x
y	Pointer to array y

Author

Xiaozhe Hu

Date

02/22/2011

Modified by Chunsheng Feng, Zheng Li on 08/29/2012

Definition at line 593 of file blas_csr.c.

10.12.2.3 fasp_blas_dcsr_add()

```
void fasp_blas_dcsr_add (
            dCSRmat * A,
            const REAL alpha,
            dCSRmat * B,
            const REAL beta,
            dCSRmat * C )
```

compute C = alpha∗A + beta∗B in CSR format

Parameters

A	Pointer to dCSRmat matrix
alpha	REAL factor alpha
B	Pointer to dCSRmat matrix
beta	REAL factor beta
C	Pointer to dCSRmat matrix

Returns

FASP_SUCCESS if succeed, ERROR if not

Author

Xiaozhe Hu

Date

11/07/2009

Modified by Chunsheng Feng, Zheng Li on 06/29/2012

Definition at line 48 of file blas_csr.c.

10.12.2.4 fasp_blas_dcsr_axm()

```
void fasp_blas_dcsr_axm (
            dCSRmat * A,
            const REAL alpha )
```

Multiply a sparse matrix A in CSR format by a scalar alpha.

Parameters

A	Pointer to dCSRmat matrix A
alpha	REAL factor alpha

Author

Chensong Zhang

Date

07/01/2009

Modified by Chunsheng Feng, Zheng Li on 06/29/2012

Definition at line 201 of file blas_csr.c.

10.12.2.5 fasp_blas_dcsr_bandwith()

```
fasp_blas_dcsr_bandwith (
            dCSRmat * A,
            INT * bndwith )
```

Get bandwith of matrix.

Parameters

A	pointer to the dCSRmat matrix
bndwith	pointer to the bandwith

Author

Zheng Li

Date

03/22/2015

Definition at line 1999 of file blas_csr.c.

10.12.2.6 fasp_blas_dcsr_mxm()

```
void fasp_blas_dcsr_mxm (
            dCSRmat * A,
            dCSRmat * B,
            dCSRmat * C )
```

Sparse matrix multiplication C=A∗B.

Parameters

A	Pointer to the dCSRmat matrix A
B	Pointer to the dCSRmat matrix B
C	Pointer to dCSRmat matrix equal to A∗B

Author

Xiaozhe Hu

Date

11/07/2009

Note

This fct will be replaced! –Chensong

Definition at line 759 of file blas_csr.c.

10.12.2.7 fasp_blas_dcsr_mxv()

```
void fasp_blas_dcsr_mxv (
            dCSRmat * A,
            REAL * x,
            REAL * y )
```

Matrix-vector multiplication y = A∗x.

Parameters

A	Pointer to dCSRmat matrix A
x	Pointer to array x
y	Pointer to array y

Author

Chensong Zhang

Date

07/01/2009

Modified by Chunsheng Feng, Xiaoqiang Yue on 05/26/2012

Definition at line 225 of file blas_csr.c.

10.12.2.8 fasp_blas_dcsr_mxv_agg()

```
void fasp_blas_dcsr_mxv_agg (
            dCSRmat * A,
            REAL * x,
            REAL * y )
```

Matrix-vector multiplication y = A∗x, where the entries of A are all ones.

Parameters

A	Pointer to dCSRmat matrix A
x	Pointer to array x
y	Pointer to array y

Author

Xiaozhe Hu

Date

02/22/2011

Modified by Chunsheng Feng, Zheng Li on 08/29/2012

Definition at line 423 of file blas_csr.c.

10.12.2.9 fasp_blas_dcsr_ptap()

```
void fasp_blas_dcsr_ptap (
            dCSRmat * Pt,
            dCSRmat * A,
            dCSRmat * P,
            dCSRmat * Ac )
```

Triple sparse matrix multiplication B=P'∗A∗P.

Parameters

Pt	Pointer to the restriction matrix
A	Pointer to the fine coefficient matrix
P	Pointer to the prolongation matrix
Ac	Pointer to the coarse coefficient matrix (output)

Author

Ludmil Zikatanov, Chensong Zhang

Date

05/10/2010

Modified by Chunsheng Feng, Zheng Li on 10/19/2012

Note

Driver to compute triple matrix product P'∗A∗P using ltz CSR format. In ltx format: ia[0]=1, ja[0] and a[0] are used as usual. When called from Fortran, ia[0], ja[0] and a[0] will be just ia(1),ja(1),a(1). For the indices, ia_ltz[k] = ia_usual[k]+1, ja_ltz[k] = ja_usual[k]+1, a_ltz[k] = a_usual[k].

Definition at line 1596 of file blas_csr.c.

10.12.2.10 fasp_blas_dcsr_rap()

```
void fasp_blas_dcsr_rap (
            dCSRmat * R,
            dCSRmat * A,
            dCSRmat * P,
            dCSRmat * RAP )
```

Triple sparse matrix multiplication B=R∗A∗P.

Parameters

R	Pointer to the dCSRmat matrix R
A	Pointer to the dCSRmat matrix A
P	Pointer to the dCSRmat matrix P
RAP	Pointer to dCSRmat matrix equal to R∗A∗P

Author

Xuehai Huang, Chensong Zhang

Date

05/10/2010

Modified by Chunsheng Feng, Xiaoqiang Yue on 05/26/2012

Note

Ref. R.E. Bank and C.C. Douglas. SMMP: Sparse Matrix Multiplication Package. Advances in Computational Mathematics, 1 (1993), pp. 127-137.

Definition at line 866 of file blas_csr.c.

10.12.2.11 fasp_blas_dcsr_rap4()

```
void fasp_blas_dcsr_rap4 (
            dCSRmat * R,
            dCSRmat * A,
            dCSRmat * P,
            dCSRmat * B,
            INT * icor_ysk )
```

Triple sparse matrix multiplication B=R∗A∗P.

Parameters

R	pointer to the dCSRmat matrix
A	pointer to the dCSRmat matrix
P	pointer to the dCSRmat matrix
B	pointer to dCSRmat matrix equal to R∗A∗P
icor_ysk	pointer to the array

Author

Feng Chunsheng, Yue Xiaoqiang

Date

08/02/2011

Note

Ref. R.E. Bank and C.C. Douglas. SMMP: Sparse Matrix Multiplication Package. Advances in Computational Mathematics, 1 (1993), pp. 127-137.

Definition at line 1698 of file blas_csr.c.

10.12.2.12 fasp_blas_dcsr_rap_agg()

```
void fasp_blas_dcsr_rap_agg (
            dCSRmat * R,
            dCSRmat * A,
            dCSRmat * P,
            dCSRmat * RAP )
```

Triple sparse matrix multiplication B=R∗A∗P.

Parameters

R	Pointer to the dCSRmat matrix R
A	Pointer to the dCSRmat matrix A
P	Pointer to the dCSRmat matrix P
RAP	Pointer to dCSRmat matrix equal to R∗A∗P

Author

Xiaozhe Hu

Date

05/10/2010

Modified by Chunsheng Feng, Xiaoqiang Yue on 05/26/2012

Note

Ref. R.E. Bank and C.C. Douglas. SMMP: Sparse Matrix Multiplication Package. Advances in Computational Mathematics, 1 (1993), pp. 127-137.

Definition at line 1148 of file blas_csr.c.

10.12.2.13 fasp_blas_dcsr_rap_agg1()

```
void fasp_blas_dcsr_rap_agg1 (
            dCSRmat * R,
            dCSRmat * A,
            dCSRmat * P,
            dCSRmat * B )
```

Triple sparse matrix multiplication B=R∗A∗P (nonzero entries of R and P are ones)

Parameters

R	Pointer to the dCSRmat matrix R
A	Pointer to the dCSRmat matrix A
P	Pointer to the dCSRmat matrix P
B	Pointer to dCSRmat matrix equal to R∗A∗P

Author

Xiaozhe Hu

Date

02/21/2011

Note

Ref. R.E. Bank and C.C. Douglas. SMMP: Sparse Matrix Multiplication Package. Advances in Computational Mathematics, 1 (1993), pp. 127-137.

Definition at line 1413 of file blas_csr.c.

10.12.2.14 fasp_blas_dcsr_vmv()

```
REAL fasp_blas_dcsr_vmv (
            dCSRmat * A,
            REAL * x,
            REAL * y )
```

vector-Matrix-vector multiplication alpha = y'∗A∗x

Parameters

A	Pointer to dCSRmat matrix A
x	Pointer to array x
y	Pointer to array y

Author

Chensong Zhang

Date

07/01/2009

Definition at line 704 of file blas_csr.c.

10.13 blas_csrl.c File Reference

BLAS2 operations for dCSRLmat matrices.

```
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_blas_dcsrl_mxv (dCSRLmat ∗A, REAL ∗x, REAL ∗y)

  Compute y = A∗x for a sparse matrix in CSRL format.

10.13.1 Detailed Description

BLAS2 operations for dCSRLmat matrices.

Note

For details of CSRL format, refer to "Optimizaing sparse matrix vector product computations using unroll and jam" by John Mellor-Crummey and John Garvin, Tech Report Rice Univ, Aug 2002.

10.13.2 Function Documentation

10.13.2.1 fasp_blas_dcsrl_mxv()

```
void fasp_blas_dcsrl_mxv (
            dCSRLmat * A,
            REAL * x,
            REAL * y )
```

Compute y = A∗x for a sparse matrix in CSRL format.

Parameters

A	Pointer to dCSRLmat matrix A
x	Pointer to REAL array of vector x
y	Pointer to REAL array of vector y

Date

2011/01/07

Definition at line 28 of file blas_csrl.c.

10.14 blas_smat.c File Reference

BLAS2 operations for *small* dense matrices.

```
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_blas_smat_axm (REAL ∗a, const INT n, const REAL alpha)

    Compute alpha∗a, store in a.
- void fasp_blas_smat_add (REAL ∗a, REAL ∗b, const INT n, const REAL alpha, const REAL beta, REAL ∗c)

    Compute c = alpha∗a + beta∗b.
- void fasp_blas_smat_mxv_nc2 (REAL ∗a, REAL ∗b, REAL ∗c)

    Compute the product of a 2∗2 matrix a and a array b, stored in c.
- void fasp_blas_smat_mxv_nc3 (REAL ∗a, REAL ∗b, REAL ∗c)

    Compute the product of a 3∗3 matrix a and a array b, stored in c.
- void fasp_blas_smat_mxv_nc5 (REAL ∗a, REAL ∗b, REAL ∗c)

    Compute the product of a 5∗5 matrix a and a array b, stored in c.
- void fasp_blas_smat_mxv_nc7 (REAL ∗a, REAL ∗b, REAL ∗c)

    Compute the product of a 7∗7 matrix a and a array b, stored in c.
- void fasp_blas_smat_mxv (REAL ∗a, REAL ∗b, REAL ∗c, const INT n)

    Compute the product of a small full matrix a and a array b, stored in c.
- void fasp_blas_smat_mul_nc2 (REAL ∗a, REAL ∗b, REAL ∗c)

    Compute the matrix product of two 2∗ matrices a and b, stored in c.
- void fasp_blas_smat_mul_nc3 (REAL ∗a, REAL ∗b, REAL ∗c)

    Compute the matrix product of two 3∗3 matrices a and b, stored in c.
- void fasp_blas_smat_mul_nc5 (REAL ∗a, REAL ∗b, REAL ∗c)

    Compute the matrix product of two 5∗5 matrices a and b, stored in c.
- void fasp_blas_smat_mul_nc7 (REAL ∗a, REAL ∗b, REAL ∗c)

    Compute the matrix product of two 7∗7 matrices a and b, stored in c.
- void fasp_blas_smat_mul (REAL ∗a, REAL ∗b, REAL ∗c, const INT n)

    Compute the matrix product of two small full matrices a and b, stored in c.
- void fasp_blas_array_axpyz_nc2 (const REAL a, REAL ∗x, REAL ∗y, REAL ∗z)

    z = a∗x + y
- void fasp_blas_array_axpyz_nc3 (const REAL a, REAL ∗x, REAL ∗y, REAL ∗z)

    z = a∗x + y
- void fasp_blas_array_axpyz_nc5 (const REAL a, REAL ∗x, REAL ∗y, REAL ∗z)

    z = a∗x + y
- void fasp_blas_array_axpyz_nc7 (const REAL a, REAL ∗x, REAL ∗y, REAL ∗z)

    z = a∗x + y
- void fasp_blas_array_axpy_nc2 (const REAL a, REAL ∗x, REAL ∗y)

    y = a∗x + y, the length of x and y is 2
- void fasp_blas_array_axpy_nc3 (const REAL a, REAL ∗x, REAL ∗y)

    y = a∗x + y, the length of x and y is 3
- void fasp_blas_array_axpy_nc5 (const REAL a, REAL ∗x, REAL ∗y)

    y = a∗x + y, the length of x and y is 5
- void fasp_blas_array_axpy_nc7 (const REAL a, REAL ∗x, REAL ∗y)

    y = a∗x + y, the length of x and y is 7
- void fasp_blas_smat_ypAx_nc2 (REAL ∗A, REAL ∗x, REAL ∗y)

    Compute y := y + Ax, where 'A' is a 2∗2 dense matrix.
- void fasp_blas_smat_ypAx_nc3 (REAL ∗A, REAL ∗x, REAL ∗y)

    Compute y := y + Ax, where 'A' is a 3∗3 dense matrix.
- void fasp_blas_smat_ypAx_nc5 (REAL ∗A, REAL ∗x, REAL ∗y)

Compute y := y + Ax, where 'A' is a 5∗5 dense matrix.
- void fasp_blas_smat_ypAx_nc7 (REAL ∗A, REAL ∗x, REAL ∗y)

  Compute y := y + Ax, where 'A' is a 7∗7 dense matrix.
- void fasp_blas_smat_ypAx (REAL ∗A, REAL ∗x, REAL ∗y, const INT n)

  Compute y := y + Ax, where 'A' is a n∗n dense matrix.
- void fasp_blas_smat_ymAx_nc2 (REAL ∗A, REAL ∗x, REAL ∗y)

  Compute y := y - Ax, where 'A' is a n∗n dense matrix.
- void fasp_blas_smat_ymAx_nc3 (REAL ∗A, REAL ∗x, REAL ∗y)

  Compute y := y - Ax, where 'A' is a n∗n dense matrix.
- void fasp_blas_smat_ymAx_nc5 (REAL ∗A, REAL ∗x, REAL ∗y)

  Compute y := y - Ax, where 'A' is a n∗n dense matrix.
- void fasp_blas_smat_ymAx_nc7 (REAL ∗A, REAL ∗x, REAL ∗y)

  Compute y := y - Ax, where 'A' is a 7∗7 dense matrix.
- void fasp_blas_smat_ymAx (REAL ∗A, REAL ∗x, REAL ∗y, const INT n)

  Compute y := y - Ax, where 'A' is a n∗n dense matrix.
- void fasp_blas_smat_aAxpby (const REAL alpha, REAL ∗A, REAL ∗x, const REAL beta, REAL ∗y, const INT n)

  Compute y:=alpha∗A∗x + beta∗y.
- void fasp_blas_smat_ymAx_ns2 (REAL ∗A, REAL ∗x, REAL ∗y)

  Compute ys := ys - Ass∗xs, where 'A' is a 2∗2 dense matrix, Ass is its saturaton part 1∗1.
- void fasp_blas_smat_ymAx_ns3 (REAL ∗A, REAL ∗x, REAL ∗y)

  Compute ys := ys - Ass∗xs, where 'A' is a 3∗3 dense matrix, Ass is its saturaton part 2∗2.
- void fasp_blas_smat_ymAx_ns5 (REAL ∗A, REAL ∗x, REAL ∗y)

  Compute ys := ys - Ass∗xs, where 'A' is a 5∗5 dense matrix, Ass is its saturaton part 4∗4.
- void fasp_blas_smat_ymAx_ns7 (REAL ∗A, REAL ∗x, REAL ∗y)

  Compute ys := ys - Ass∗xs, where 'A' is a 7∗7 dense matrix, Ass is its saturaton part 6∗6.
- void fasp_blas_smat_ymAx_ns (REAL ∗A, REAL ∗x, REAL ∗y, const INT n)

  Compute ys := ys - Ass∗xs, where 'A' is a n∗n dense matrix, Ass is its saturaton part (n-1)∗(n-1).

10.14.1 Detailed Description

BLAS2 operations for *small* dense matrices.

Warning

The rountines are designed for full matrices only!

10.14.2 Function Documentation

10.14.2.1 fasp_blas_array_axpy_nc2()

```
void fasp_blas_array_axpy_nc2 (
            const REAL a,
            REAL * x,
            REAL * y )
```

y = a∗x + y, the length of x and y is 2

Parameters

a	REAL factor a
x	Pointer to the original array
y	Pointer to the destination array

Author

Xiaozhe Hu

Date

18/11/2011

Definition at line 686 of file blas_smat.c.

10.14.2.2 fasp_blas_array_axpy_nc3()

```
void fasp_blas_array_axpy_nc3 (
            const REAL a,
            REAL * x,
            REAL * y )
```

y = a∗x + y, the length of x and y is 3

Parameters

a	REAL factor a
x	Pointer to the original array
y	Pointer to the destination array

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 709 of file blas_smat.c.

10.14.2.3 fasp_blas_array_axpy_nc5()

```
void fasp_blas_array_axpy_nc5 (
            const REAL a,
```

```
REAL * x,
REAL * y )
```

y = a∗x + y, the length of x and y is 5

Parameters

a	REAL factor a
x	Pointer to the original array
y	Pointer to the destination array

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 738 of file blas_smat.c.

10.14.2.4 fasp_blas_array_axpy_nc7()

```
void fasp_blas_array_axpy_nc7 (
            const REAL a,
            REAL * x,
            REAL * y )
```

y = a∗x + y, the length of x and y is 7

Parameters

a	REAL factor a
x	Pointer to the original array
y	Pointer to the destination array

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 785 of file blas_smat.c.

10.14.2.5 fasp_blas_array_axpyz_nc2()

```
void fasp_blas_array_axpyz_nc2 (
            const REAL a,
```

```
            REAL * x,
            REAL * y,
            REAL * z )
```

z = a∗x + y

Parameters

a	REAL factor a
x	Pointer to the original array 1
y	Pointer to the original array 2
z	Pointer to the destination array

Author

Xiaozhe Hu

Date

18/11/2011

Note

z is the third array and the length of x, y and z is 2

Definition at line 501 of file blas_smat.c.

10.14.2.6 fasp_blas_array_axpyz_nc3()

```
void fasp_blas_array_axpyz_nc3 (
            const REAL a,
            REAL * x,
            REAL * y,
            REAL * z )
```

z = a∗x + y

Parameters

a	REAL factor a
x	Pointer to the original array 1
y	Pointer to the original array 2
z	Pointer to the destination array

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Note

z is the third array and the length of x, y and z is 3

Definition at line 528 of file blas_smat.c.

10.14.2.7 fasp_blas_array_axpyz_nc5()

```
void fasp_blas_array_axpyz_nc5 (
            const REAL a,
            REAL * x,
            REAL * y,
            REAL * z )
```

z = a∗x + y

Parameters

a	REAL factor a
x	Pointer to the original array 1
y	Pointer to the original array 2
z	Pointer to the destination array

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Note

z is the third array and the length of x, y and z is 5

Definition at line 561 of file blas_smat.c.

10.14.2.8 fasp_blas_array_axpyz_nc7()

```
void fasp_blas_array_axpyz_nc7 (
            const REAL a,
            REAL * x,
            REAL * y,
            REAL * z )
```

z = a∗x + y

Parameters

a	REAL factor a
x	Pointer to the original array 1
y	Pointer to the original array 2
z	Pointer to the destination array

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Note

z is the third array and the length of x, y and z is 7

Definition at line 612 of file blas_smat.c.

10.14.2.9 fasp_blas_smat_aAxpby()

```
void fasp_blas_smat_aAxpby (
            const REAL alpha,
            REAL * A,
            REAL * x,
            const REAL beta,
            REAL * y,
            const INT n )
```

Compute y:=alpha∗A∗x + beta∗y.

Parameters

alpha	REAL factor alpha
A	Pointer to the REAL array which stands for a n∗n full matrix
x	Pointer to the REAL array with length n
beta	REAL factor beta
y	Pointer to the REAL array with length n
n	Length of array x and y

Author

Zhiyang Zhou

Date

2010/10/25

Definition at line 1309 of file blas_smat.c.

10.14.2.10 fasp_blas_smat_add()

```
void fasp_blas_smat_add (
            REAL * a,
            REAL * b,
            const INT n,
            const REAL alpha,
            const REAL beta,
            REAL * c )
```

Compute c = alpha∗a + beta∗b.

Parameters

a	Pointer to the REAL array which stands a n∗n matrix
b	Pointer to the REAL array which stands a n∗n matrix
n	Dimension of the matrix
alpha	Scalar
beta	Scalar
c	Pointer to the REAL array which stands a n∗n matrix

Author

Xiaozhe Hu

Date

05/26/2014

Definition at line 55 of file blas_smat.c.

10.14.2.11 fasp_blas_smat_axm()

```
void fasp_blas_smat_axm (
            REAL * a,
            const INT n,
            const REAL alpha )
```

Compute alpha∗a, store in a.

Parameters

a	Pointer to the REAL array which stands a n∗n matrix
n	Dimension of the matrix
alpha	Scalar

Author

Xiaozhe Hu

Date

05/26/2014

Definition at line 26 of file blas_smat.c.

10.14.2.12 fasp_blas_smat_mul()

```
void fasp_blas_smat_mul (
            REAL * a,
            REAL * b,
            REAL * c,
            const INT n )
```

Compute the matrix product of two small full matrices a and b, stored in c.

Parameters

a	Pointer to the REAL array which stands a n∗n matrix
b	Pointer to the REAL array which stands a n∗n matrix
c	Pointer to the REAL array which stands a n∗n matrix
n	Dimension of the matrix

Author

Xiaozhe Hu, Shiquan Zhang

Date

04/21/2010

Definition at line 449 of file blas_smat.c.

10.14.2.13 fasp_blas_smat_mul_nc2()

```
void fasp_blas_smat_mul_nc2 (
            REAL * a,
            REAL * b,
            REAL * c )
```

Compute the matrix product of two 2∗ matrices a and b, stored in c.

Parameters

a	Pointer to the REAL array which stands a n∗n matrix
b	Pointer to the REAL array which stands a n∗n matrix
c	Pointer to the REAL array which stands a n∗n matrix

Author

Xiaozhe Hu

Date

18/11/2011

Definition at line 234 of file blas_smat.c.

10.14.2.14 fasp_blas_smat_mul_nc3()

```
void fasp_blas_smat_mul_nc3 (
            REAL * a,
            REAL * b,
            REAL * c )
```

Compute the matrix product of two 3∗3 matrices a and b, stored in c.

Parameters

a	Pointer to the REAL array which stands a n∗n matrix
b	Pointer to the REAL array which stands a n∗n matrix
c	Pointer to the REAL array which stands a n∗n matrix

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 263 of file blas_smat.c.

10.14.2.15 fasp_blas_smat_mul_nc5()

```
void fasp_blas_smat_mul_nc5 (
            REAL * a,
            REAL * b,
            REAL * c )
```

Compute the matrix product of two 5∗5 matrices a and b, stored in c.

Parameters

a	Pointer to the REAL array which stands a 5∗5 matrix
b	Pointer to the REAL array which stands a 5∗5 matrix
c	Pointer to the REAL array which stands a 5∗5 matrix

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 300 of file blas_smat.c.

10.14.2.16 fasp_blas_smat_mul_nc7()

```
void fasp_blas_smat_mul_nc7 (
            REAL * a,
            REAL * b,
            REAL * c )
```

Compute the matrix product of two 7∗7 matrices a and b, stored in c.

Parameters

a	Pointer to the REAL array which stands a 7∗7 matrix
b	Pointer to the REAL array which stands a 7∗7 matrix
c	Pointer to the REAL array which stands a 7∗7 matrix

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 359 of file blas_smat.c.

10.14.2.17 fasp_blas_smat_mxv()

```
void fasp_blas_smat_mxv (
            REAL * a,
            REAL * b,
            REAL * c,
            const INT n )
```

Compute the product of a small full matrix a and a array b, stored in c.

Parameters

a	Pointer to the REAL array which stands a n∗n matrix
b	Pointer to the REAL array with length n
c	Pointer to the REAL array with length n
n	Dimension of the matrix

Author

Xiaozhe Hu, Shiquan Zhang

Date

04/21/2010

Definition at line 184 of file blas_smat.c.

10.14.2.18 fasp_blas_smat_mxv_nc2()

```
void fasp_blas_smat_mxv_nc2 (
            REAL * a,
            REAL * b,
            REAL * c )
```

Compute the product of a 2∗2 matrix a and a array b, stored in c.

Parameters

a	Pointer to the REAL array which stands a 2*2 matrix
b	Pointer to the REAL array with length 2
c	Pointer to the REAL array with length 2

Author

Xiaozhe Hu

Date

18/11/2010

Definition at line 84 of file blas_smat.c.

10.14.2.19 fasp_blas_smat_mxv_nc3()

```
void fasp_blas_smat_mxv_nc3 (
            REAL * a,
            REAL * b,
            REAL * c )
```

Compute the product of a 3*3 matrix a and a array b, stored in c.

Parameters

a	Pointer to the REAL array which stands a 3*3 matrix
b	Pointer to the REAL array with length 3
c	Pointer to the REAL array with length 3

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 106 of file blas_smat.c.

10.14.2.20 fasp_blas_smat_mxv_nc5()

```
void fasp_blas_smat_mxv_nc5 (
            REAL * a,
```

```
REAL * b,
REAL * c )
```

Compute the product of a 5*5 matrix a and a array b, stored in c.

Parameters

a	Pointer to the REAL array which stands a 5∗5 matrix
b	Pointer to the REAL array with length 5
c	Pointer to the REAL array with length 5

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 129 of file blas_smat.c.

10.14.2.21 fasp_blas_smat_mxv_nc7()

```
void fasp_blas_smat_mxv_nc7 (
            REAL * a,
            REAL * b,
            REAL * c )
```

Compute the product of a 7∗7 matrix a and a array b, stored in c.

Parameters

a	Pointer to the REAL array which stands a 7∗7 matrix
b	Pointer to the REAL array with length 7
c	Pointer to the REAL array with length 7

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 155 of file blas_smat.c.

10.14.2.22 fasp_blas_smat_ymAx()

```
void fasp_blas_smat_ymAx (
            REAL * A,
            REAL * x,
            REAL * y,
            const INT n )
```

Compute y := y - Ax, where 'A' is a n∗n dense matrix.

Parameters

A	Pointer to the n∗n dense matrix
x	Pointer to the REAL array with length n
y	Pointer to the REAL array with length n
n	the dimension of the dense matrix

Author

Zhiyang Zhou, Xiaozhe Hu

Date

2010/10/25

Definition at line 1208 of file blas_smat.c.

10.14.2.23 fasp_blas_smat_ymAx_nc2()

```
void fasp_blas_smat_ymAx_nc2 (
            REAL * A,
            REAL * x,
            REAL * y )
```

Compute y := y - Ax, where 'A' is a n∗n dense matrix.

Parameters

A	Pointer to the 2∗2 dense matrix
x	Pointer to the REAL array with length 3
y	Pointer to the REAL array with length 3

Author

Xiaozhe Hu

Date

18/11/2011

Note

Works for 2-component

Definition at line 1078 of file blas_smat.c.

10.14.2.24 fasp_blas_smat_ymAx_nc3()

```
void fasp_blas_smat_ymAx_nc3 (
            REAL * A,
            REAL * x,
            REAL * y )
```

Compute y := y - Ax, where 'A' is a n∗n dense matrix.

Parameters

A	Pointer to the 3∗3 dense matrix
x	Pointer to the REAL array with length 3
y	Pointer to the REAL array with length 3

Author

Xiaozhe Hu, Zhiyang Zhou

Date

01/06/2011

Note

Works for 3-component

Definition at line 1106 of file blas_smat.c.

10.14.2.25 fasp_blas_smat_ymAx_nc5()

```
void fasp_blas_smat_ymAx_nc5 (
            REAL * A,
            REAL * x,
            REAL * y )
```

Compute y := y - Ax, where 'A' is a n∗n dense matrix.

Parameters

A	Pointer to the 5∗5 dense matrix
x	Pointer to the REAL array with length 5
y	Pointer to the REAL array with length 5

Author

Xiaozhe Hu, Zhiyang Zhou

Date

01/06/2011

Note

Works for 5-component

Definition at line 1136 of file blas_smat.c.

10.14.2.26 fasp_blas_smat_ymAx_nc7()

```
void fasp_blas_smat_ymAx_nc7 (
            REAL * A,
            REAL * x,
            REAL * y )
```

Compute y := y - Ax, where 'A' is a 7∗7 dense matrix.

Parameters

A	Pointer to the 7∗7 dense matrix
x	Pointer to the REAL array with length 7
y	Pointer to the REAL array with length 7

Author

Xiaozhe Hu, Zhiyang Zhou

Date

01/06/2011

Note

Works for 7-component

Definition at line 1170 of file blas_smat.c.

10.14.2.27 fasp_blas_smat_ymAx_ns()

```
void fasp_blas_smat_ymAx_ns (
            REAL * A,
            REAL * x,
            REAL * y,
            const INT n )
```

Compute ys := ys - Ass∗xs, where 'A' is a n∗n dense matrix, Ass is its saturaton part (n-1)∗(n-1).

Parameters

A	Pointer to the n∗n dense matrix
x	Pointer to the REAL array with length n-1
y	Pointer to the REAL array with length n-1
n	the dimension of the dense matrix

Author

Xiaozhe Hu

Date

2010/10/25

Note

Only for block smoother for saturation block without explictly use saturation block!!

Definition at line 1483 of file blas_smat.c.

10.14.2.28 fasp_blas_smat_ymAx_ns2()

```
void fasp_blas_smat_ymAx_ns2 (
            REAL * A,
            REAL * x,
            REAL * y )
```

Compute ys := ys - Ass∗xs, where 'A' is a 2∗2 dense matrix, Ass is its saturaton part 1∗1.

Parameters

A	Pointer to the 2∗2 dense matrix
x	Pointer to the REAL array with length 1
y	Pointer to the REAL array with length 1

Author

Xiaozhe Hu

Date

2011/11/18

Note

Works for 2-component (Xiaozhe) Only for block smoother for saturation block without explictly use saturation block!!

Definition at line 1359 of file blas_smat.c.

10.14.2.29 fasp_blas_smat_ymAx_ns3()

```
void fasp_blas_smat_ymAx_ns3 (
            REAL * A,
            REAL * x,
            REAL * y )
```

Compute ys := ys - Ass∗xs, where 'A' is a 3∗3 dense matrix, Ass is its saturaton part 2∗2.

Parameters

A	Pointer to the 3∗3 dense matrix
x	Pointer to the REAL array with length 2
y	Pointer to the REAL array with length 2

Author

Xiaozhe Hu

Date

2010/10/25

Note

Works for 3-component (Xiaozhe) Only for block smoother for saturation block without explictly use saturation block!!

Definition at line 1383 of file blas_smat.c.

10.14.2.30 fasp_blas_smat_ymAx_ns5()

```
void fasp_blas_smat_ymAx_ns5 (
            REAL * A,
            REAL * x,
            REAL * y )
```

Compute ys := ys - Ass∗xs, where 'A' is a 5∗5 dense matrix, Ass is its saturaton part 4∗4.

Parameters

A	Pointer to the 5∗5 dense matrix
x	Pointer to the REAL array with length 4
y	Pointer to the REAL array with length 4

Author

Xiaozhe Hu

Date

2010/10/25

Note

Works for 5-component (Xiaozhe) Only for block smoother for saturation block without explictly use saturation block!!

Definition at line 1411 of file blas_smat.c.

10.14.2.31 fasp_blas_smat_ymAx_ns7()

```
void fasp_blas_smat_ymAx_ns7 (
            REAL * A,
            REAL * x,
            REAL * y )
```

Compute ys := ys - Ass∗xs, where 'A' is a 7∗7 dense matrix, Ass is its saturaton part 6∗6.

Parameters

A	Pointer to the 7∗7 dense matrix
x	Pointer to the REAL array with length 6
y	Pointer to the REAL array with length 6

Author

Xiaozhe Hu

Date

2010/10/25

Note

Works for 7-component (Xiaozhe) Only for block smoother for saturation block without explictly use saturation block!!

Definition at line 1445 of file blas_smat.c.

10.14.2.32 fasp_blas_smat_ypAx()

```
void fasp_blas_smat_ypAx (
            REAL * A,
            REAL * x,
            REAL * y,
            const INT n )
```

Compute y := y + Ax, where 'A' is a n∗n dense matrix.

Parameters

A	Pointer to the n∗n dense matrix
x	Pointer to the REAL array with length n
y	Pointer to the REAL array with length n
n	Dimension of the dense matrix

Author

Zhiyang Zhou

Date

2010/10/25

Definition at line 977 of file blas_smat.c.

10.14.2.33 fasp_blas_smat_ypAx_nc2()

```
void fasp_blas_smat_ypAx_nc2 (
            REAL * A,
            REAL * x,
            REAL * y )
```

Compute y := y + Ax, where 'A' is a 2∗2 dense matrix.

Parameters

A	Pointer to the 3∗3 dense matrix
x	Pointer to the REAL array with length 3
y	Pointer to the REAL array with length 3

Author

Xiaozhe Hu

Date

2011/11/18

Definition at line 858 of file blas_smat.c.

10.14.2.34 fasp_blas_smat_ypAx_nc3()

```
void fasp_blas_smat_ypAx_nc3 (
            REAL * A,
            REAL * x,
            REAL * y )
```

Compute y := y + Ax, where 'A' is a 3∗3 dense matrix.

Parameters

A	Pointer to the 3∗3 dense matrix
x	Pointer to the REAL array with length 3
y	Pointer to the REAL array with length 3

Author

Zhiyang Zhou, Xiaozhe Hu

Date

2010/10/25

Definition at line 884 of file blas_smat.c.

10.14.2.35 fasp_blas_smat_ypAx_nc5()

```
void fasp_blas_smat_ypAx_nc5 (
            REAL * A,
```

```
REAL * x,
REAL * y )
```

Compute y := y + Ax, where 'A' is a 5∗5 dense matrix.

Parameters

A	Pointer to the 5∗5 dense matrix
x	Pointer to the REAL array with length 5
y	Pointer to the REAL array with length 5

Author

Zhiyang Zhou, Xiaozhe Hu

Date

2010/10/25

Definition at line 911 of file blas_smat.c.

10.14.2.36 fasp_blas_smat_ypAx_nc7()

```
void fasp_blas_smat_ypAx_nc7 (
            REAL * A,
            REAL * x,
            REAL * y )
```

Compute y := y + Ax, where 'A' is a 7∗7 dense matrix.

Parameters

A	Pointer to the 7∗7 dense matrix
x	Pointer to the REAL array with length 7
y	Pointer to the REAL array with length 7

Author

Zhiyang Zhou, Xiaozhe Hu

Date

2010/10/25

Definition at line 942 of file blas_smat.c.

10.15 blas_str.c File Reference

BLAS2 operations for dSTRmat matrices.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_blas_dstr_aAxpy (const REAL alpha, dSTRmat *A, REAL *x, REAL *y)

  *Matrix-vector multiplication y = alpha*A*x + y.*
- void fasp_blas_dstr_mxv (dSTRmat *A, REAL *x, REAL *y)

  *Matrix-vector multiplication y = A*x.*
- INT fasp_dstr_diagscale (dSTRmat *A, dSTRmat *B)

  B=D^{-1}A.

10.15.1 Detailed Description

BLAS2 operations for dSTRmat matrices.

10.15.2 Function Documentation

10.15.2.1 fasp_blas_dstr_aAxpy()

```
void fasp_blas_dstr_aAxpy (
            const REAL alpha,
            dSTRmat * A,
            REAL * x,
            REAL * y )
```

Matrix-vector multiplication y = alpha*A*x + y.

Parameters

alpha	REAL factor alpha
A	Pointer to dSTRmat matrix
x	Pointer to REAL array
y	Pointer to REAL array

Author

Zhiyang Zhou, Xiaozhe Hu, Shiquan Zhang

Date

2010/10/15

Definition at line 47 of file blas_str.c.

10.15.2.2 fasp_blas_dstr_mxv()

```
void fasp_blas_dstr_mxv (
            dSTRmat * A,
            REAL * x,
            REAL * y )
```

Matrix-vector multiplication y = A*x.

Parameters

A	Pointer to dSTRmat matrix
x	Pointer to REAL array
y	Pointer to REAL array

Author

Chensong Zhang

Date

04/27/2013

Definition at line 117 of file blas_str.c.

10.15.2.3 fasp_dstr_diagscale()

```
INT fasp_dstr_diagscale (
            dSTRmat * A,
            dSTRmat * B )
```

B=D^{-1}A.

Parameters

A	Pointer to a 'dSTRmat' type matrix A
B	Pointer to a 'dSTRmat' type matrix B

Author

Shiquan Zhang

Date

2010/10/15

Modified by Chunsheng Feng, Zheng Li

Date

08/30/2012

Definition at line 142 of file blas_str.c.

10.16 blas_vec.c File Reference

BLAS1 operations for vectors.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_blas_dvec_axpy (const REAL a, dvector ∗x, dvector ∗y)

  y = a∗x + y
- void fasp_blas_dvec_axpyz (const REAL a, dvector ∗x, dvector ∗y, dvector ∗z)

  z = a∗x + y, z is a third vector (z is cleared)
- REAL fasp_blas_dvec_dotprod (dvector ∗x, dvector ∗y)

  Inner product of two vectors (x,y)
- REAL fasp_blas_dvec_relerr (dvector ∗x, dvector ∗y)

  Relative error of two dvector x and y.
- REAL fasp_blas_dvec_norm1 (dvector ∗x)

  L1 norm of dvector x.
- REAL fasp_blas_dvec_norm2 (dvector ∗x)

  L2 norm of dvector x.
- REAL fasp_blas_dvec_norminf (dvector ∗x)

  Linf norm of dvector x.

10.16.1 Detailed Description

BLAS1 operations for vectors.

10.16.2 Function Documentation

10.16.2.1 fasp_blas_dvec_axpy()

```
void fasp_blas_dvec_axpy (
            const REAL a,
            dvector * x,
            dvector * y )
```

y = a∗x + y

Parameters

a	REAL factor a
x	Pointer to dvector x
y	Pointer to dvector y

Author

Chensong Zhang

Date

07/01/209

Modified by Chunsheng Feng, Xiaoqiang Yue on 05/23/2012

Definition at line 33 of file blas_vec.c.

10.16.2.2 fasp_blas_dvec_axpyz()

```
void fasp_blas_dvec_axpyz (
            const REAL a,
            dvector * x,
            dvector * y,
            dvector * z )
```

z = a∗x + y, z is a third vector (z is cleared)

Parameters

a	REAL factor a
x	Pointer to dvector x
y	Pointer to dvector y
z	Pointer to dvector z

Author

Chensong Zhang

Date

07/01/209

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

05/23/2012

Definition at line 85 of file blas_vec.c.

10.16.2.3 fasp_blas_dvec_dotprod()

```
REAL fasp_blas_dvec_dotprod (
            dvector * x,
            dvector * y )
```

Inner product of two vectors (x,y)

Parameters

x	Pointer to dvector x
y	Pointer to dvector y

Returns

Inner product

Author

Chensong Zhang

Date

07/01/209

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

05/23/2012

Definition at line 121 of file blas_vec.c.

10.16.2.4 fasp_blas_dvec_norm1()

```
REAL fasp_blas_dvec_norm1 (
            dvector * x )
```

L1 norm of dvector x.

Parameters

x	Pointer to dvector x

Returns

L1 norm of x

Author

Chensong Zhang

Date

07/01/209

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

05/23/2012

Definition at line 222 of file blas_vec.c.

10.16.2.5 fasp_blas_dvec_norm2()

```
REAL fasp_blas_dvec_norm2 (
            dvector * x )
```

L2 norm of dvector x.

Parameters

x	Pointer to dvector x

Returns

L2 norm of x

Author

Chensong Zhang

Date

07/01/209

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

05/23/2012

Definition at line 265 of file blas_vec.c.

10.16.2.6 fasp_blas_dvec_norminf()

```
REAL fasp_blas_dvec_norminf (
            dvector * x )
```

Linf norm of dvector x.

Parameters

x	Pointer to dvector x

Returns

L_inf norm of x

Author

Chensong Zhang

Date

07/01/209

Definition at line 305 of file blas_vec.c.

10.16.2.7 fasp_blas_dvec_relerr()

```
REAL fasp_blas_dvec_relerr (
            dvector * x,
            dvector * y )
```

Relative error of two dvector x and y.

Parameters

x	Pointer to dvector x
y	Pointer to dvector y

Returns

relative error ||x-y||/||x||

Author

Chensong Zhang

Date

07/01/209

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

05/23/2012

Definition at line 167 of file blas_vec.c.

10.17 checkmat.c File Reference

Check matrix properties.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- INT fasp_check_diagpos (dCSRmat ∗A)

  Check positivity of diagonal entries of a CSR sparse matrix.
- SHORT fasp_check_diagzero (dCSRmat ∗A)

  Check wether a CSR sparse matrix has diagonal entries that are very close to zero.
- INT fasp_check_diagdom (dCSRmat ∗A)

  Check whether a matrix is diagonal dominant.
- INT fasp_check_symm (dCSRmat ∗A)

  Check symmetry of a sparse matrix of CSR format.
- SHORT fasp_check_dCSRmat (dCSRmat ∗A)

  Check whether an dCSRmat matrix is valid or not.
- SHORT fasp_check_iCSRmat (iCSRmat ∗A)

  Check whether an iCSRmat matrix is valid or not.

10.17.1 Detailed Description

Check matrix properties.

10.17.2 Function Documentation

10.17.2.1 fasp_check_dCSRmat()

```
SHORT fasp_check_dCSRmat (
            dCSRmat * A )
```

Check whether an dCSRmat matrix is valid or not.

Parameters

A	Pointer to the matrix in dCSRmat format

Author

Shuo Zhang

Date

03/29/2009

Definition at line 275 of file checkmat.c.

10.17.2.2 fasp_check_diagdom()

```
INT fasp_check_diagdom (
            dCSRmat * A )
```

Check whether a matrix is diagonal dominant.

INT fasp_check_diagdom (dCSRmat *A)

Parameters

A	Pointer to the dCSRmat matrix

Returns

Number of the rows which are diagonal dominant

Note

The routine chechs whether the sparse matrix is diagonal dominant on every row. It will print out the percentage of the rows which are diagonal dominant and which are not; the routine will return the number of the rows which are diagonal dominant.

Author

Shuo Zhang

Date

03/29/2009

Definition at line 108 of file checkmat.c.

10.17.2.3 fasp_check_diagpos()

```
INT fasp_check_diagpos (
            dCSRmat * A )
```

Check positivity of diagonal entries of a CSR sparse matrix.

Parameters

A	Pointer to dCSRmat matrix

Returns

Number of negative diagonal entries

Author

Shuo Zhang

Date

03/29/2009

Definition at line 27 of file checkmat.c.

10.17.2.4 fasp_check_diagzero()

```
SHORT fasp_check_diagzero (
            dCSRmat * A )
```

Check wether a CSR sparse matrix has diagonal entries that are very close to zero.

Parameters

A	pointr to the dCSRmat matrix

Returns

FASP_SUCCESS if no diagonal entry is clase to zero, else ERROR

Author

Shuo Zhang

Date

03/29/2009

Definition at line 64 of file checkmat.c.

10.17.2.5 fasp_check_iCSRmat()

```
SHORT fasp_check_iCSRmat (
            iCSRmat * A )
```

Check whether an iCSRmat matrix is valid or not.

Parameters

A	Pointer to the matrix in iCSRmat format

Author

Shuo Zhang

Date

03/29/2009

Definition at line 309 of file checkmat.c.

10.17.2.6 fasp_check_symm()

```
INT fasp_check_symm (
            dCSRmat * A )
```

Check symmetry of a sparse matrix of CSR format.

Parameters

A	Pointer to the dCSRmat matrix

Returns

1 and 2 if the structure of the matrix is not symmetric; 0 if the structure of the matrix is symmetric,

Note

Print the maximal relative difference between matrix and its transpose.

Author

Shuo Zhang

Date

03/29/2009

Definition at line 153 of file checkmat.c.

10.18 coarsening_cr.c File Reference

Coarsening with Brannick-Falgout strategy.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- INT fasp_amg_coarsening_cr (const INT i_0, const INT i_n, dCSRmat *A, ivector *vertices, AMG_param *param)

  CR coarsening.

10.18.1 Detailed Description

Coarsening with Brannick-Falgout strategy.

10.18.2 Function Documentation

10.18.2.1 fasp_amg_coarsening_cr()

```
INT fasp_amg_coarsening_cr (
            const INT i_0,
            const INT i_n,
            dCSRmat * A,
            ivector * vertices,
            AMG_param * param )
```

CR coarsening.

Parameters

i_0	Starting index
i_n	Ending index
A	Pointer to dCSRmat: the coefficient matrix (index starts from 0)
vertices	Pointer to CF, 0: fpt (current level) or 1: cpt
param	Pointer to AMG_param: AMG parameters

Returns

Number of coarse level points

Author

James Brannick

Date

04/21/2010

Modified by Chunsheng Feng, Zheng Li on 10/14/2012 CR STAGES

Definition at line 42 of file coarsening_cr.c.

10.19 coarsening_rs.c File Reference

Coarsening with a modified Ruge-Stuben strategy.

```
#include "fasp.h"
#include "fasp_functs.h"
#include "linklist.inl"
```

Functions

- SHORT fasp_amg_coarsening_rs (dCSRmat ∗A, ivector ∗vertices, dCSRmat ∗P, iCSRmat ∗S, AMG_param ∗param)

  Standard and aggressive coarsening schemes.

10.19.1 Detailed Description

Coarsening with a modified Ruge-Stuben strategy.

Note

Ref Multigrid by U. Trottenberg, C. W. Oosterlee and A. Schuller Appendix P475 A.7 (by A. Brandt, P. Oswald and K. Stuben) Academic Press Inc., San Diego, CA, 2001.

ATTENTION: Do NOT use auto-indentation in this file!!!

10.19.2 Function Documentation

10.19.2.1 fasp_amg_coarsening_rs()

```
SHORT fasp_amg_coarsening_rs (
            dCSRmat * A,
            ivector * vertices,
            dCSRmat * P,
            iCSRmat * S,
            AMG_param * param )
```

Standard and aggressive coarsening schemes.

Parameters

A	Pointer to dCSRmat: Coefficient matrix (index starts from 0)
vertices	Indicator vector for the C/F splitting of the variables
P	Interpolation matrix (nonzero pattern only)
S	Strong connection matrix
param	Pointer to AMG_param: AMG parameters

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Xuehai Huang, Chensong Zhang, Xiaozhe Hu, Ludmil Zikatanov

Date

09/06/2010

Note

vertices = 0: fine; 1: coarse; 2: isolated or special

Modified by Xiaozhe Hu on 05/23/2011: add strength matrix as an argument Modified by Xiaozhe Hu on 04/24/2013: modify aggressive coarsening Modified by Chensong Zhang on 04/28/2013: remove linked list Modified by Chensong Zhang on 05/11/2013: restructure the code

Definition at line 61 of file coarsening_rs.c.

10.20 convert.c File Reference

Some utilities for format conversion.

```
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- unsigned long fasp_aux_change_endian4 (unsigned long x)
  Swap order for different endian systems.
- double fasp_aux_change_endian8 (double x)
  Swap order for different endian systems.
- double fasp_aux_bbyteToldouble (unsigned char bytes[ ])
  Swap order of double-precision float for different endian systems.
- INT endian_convert_int (const INT inum, const INT ilength, const INT endianflag)
  Swap order of an INT number.
- REAL endian_convert_real (const REAL rnum, const INT vlength, const INT endianflag)
  Swap order of a REAL number.

10.20.1 Detailed Description

Some utilities for format conversion.

10.20.2 Function Documentation

10.20.2.1 endian_convert_int()

```
INT endian_convert_int (
            const INT inum,
            const INT ilength,
            const INT endianflag )
```

Swap order of an INT number.

Parameters

inum	An INT value
ilength	Length of INT: 2 for short, 4 for int, 8 for long
endianflag	If endianflag = 1, it returns inum itself If endianflag = 2, it returns the swapped inum

Returns

Value of inum or swapped inum

Author

Ziteng Wang

Date

2012-12-24

Definition at line 105 of file convert.c.

10.20.2.2 endian_convert_real()

```
REAL endian_convert_real (
            const REAL rnum,
            const INT ilength,
            const INT endianflag )
```

Swap order of a REAL number.

Parameters

rnum	An REAL value
ilength	Length of INT: 2 for short, 4 for int, 8 for long
endianflag	If endianflag = 1, it returns rnum itself If endianflag = 2, it returns the swapped rnum

Returns

Value of rnum or swapped rnum

Author

Ziteng Wang

Date

2012-12-24

Definition at line 137 of file convert.c.

10.20.2.3 fasp_aux_bbyteToldouble()

```
double fasp_aux_bbyteToldouble (
            unsigned char bytes[] )
```

Swap order of double-precision float for different endian systems.

Parameters

bytes	A unsigned char

Returns

Unsigend long ineger after swapping

Author

Chensong Zhang

Date

11/16/2009

Definition at line 74 of file convert.c.

10.20.2.4 fasp_aux_change_endian4()

```
unsigned long fasp_aux_change_endian4 (
            unsigned long x )
```

Swap order for different endian systems.

Parameters

x	An unsigned long integer

Returns

Unsigend long ineger after swapping

Author

Chensong Zhang

Date

11/16/2009

Definition at line 25 of file convert.c.

10.20.2.5 fasp_aux_change_endian8()

```
double fasp_aux_change_endian8 (
            double x )
```

Swap order for different endian systems.

Parameters

x	A unsigned long integer

Returns

Unsigend long ineger after swapping

Author

Chensong Zhang

Date

11/16/2009

Definition at line 43 of file convert.c.

10.21 doxygen.h File Reference

Main page for Doygen documentation.

10.21.1 Detailed Description

Main page for Doygen documentation.

10.22 eigen.c File Reference

Subroutines for computing the extreme eigenvalues.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- REAL fasp_dcsr_eig (dCSRmat ∗A, const REAL tol, const INT maxit)
  Approximate the largest eigenvalue of A by the power method.

10.22.1 Detailed Description

Subroutines for computing the extreme eigenvalues.

10.22.2 Function Documentation

10.22.2.1 fasp_dcsr_eig()

```
REAL fasp_dcsr_eig (
            dCSRmat * A,
            const REAL tol,
            const INT maxit )
```

Approximate the largest eigenvalue of A by the power method.

Parameters

A	Pointer to the dCSRmat matrix
tol	Tolerance for stopping the power method
maxit	Max number of iterations

Returns

Largest eigenvalue

Author

Xiaozhe Hu

Date

01/25/2011

Definition at line 29 of file eigen.c.

10.23 famg.c File Reference

Full AMG method as an iterative solver (main file)

```
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_solver_famg (dCSRmat *A, dvector *b, dvector *x, AMG_param *param)

  Solve Ax=b by full AMG.

10.23.1 Detailed Description

Full AMG method as an iterative solver (main file)

10.23.2 Function Documentation

10.23.2.1 fasp_solver_famg()

```
void fasp_solver_famg (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            AMG_param * param )
```

Solve Ax=b by full AMG.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns
param	Pointer to AMG_param: AMG parameters

Author

Xiaozhe Hu

Date

02/27/2011

Modified by Chensong Zhang on 01/10/2012 Modified by Chensong Zhang on 05/05/2013: Remove error handling for AMG setup

Definition at line 31 of file famg.c.

10.24 fasp.h File Reference

Main header file for FASP.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "fasp_const.h"
```

Data Structures

- struct ddenmat

  Dense matrix of REAL type.
- struct idenmat

  Dense matrix of INT type.
- struct dCSRmat

  Sparse matrix of REAL type in CSR format.
- struct iCSRmat

  Sparse matrix of INT type in CSR format.
- struct dCOOmat

  Sparse matrix of REAL type in COO (or IJ) format.
- struct iCOOmat

  Sparse matrix of INT type in COO (or IJ) format.
- struct dCSRLmat

Sparse matrix of REAL type in CSRL format.

- struct dSTRmat

  Structure matrix of REAL type.
- struct dvector

  Vector with n entries of REAL type.
- struct ivector

  Vector with n entries of INT type.
- struct ILU_param

  Parameters for ILU.
- struct ILU_data

  Data for ILU setup.
- struct Schwarz_param

  Parameters for Schwarz method.
- struct Mumps_data

  Parameters for MUMPS interface.
- struct Pardiso_data

  Parameters for Intel MKL PARDISO interface.
- struct Schwarz_data

  Data for Schwarz methods.
- struct AMG_param

  Parameters for AMG solver.
- struct AMG_data

  Data for AMG solvers.
- struct precond_data

  Data passed to the preconditioners.
- struct precond_data_str

  Data passed to the preconditioner for dSTRmat matrices.
- struct precond_diagstr

  Data passed to diagonal preconditioner for dSTRmat matrices.
- struct precond

  Preconditioner data and action.
- struct mxv_matfree

  Matrix-vector multiplication, replace the actual matrix.
- struct input_param

  Input parameters.
- struct itsolver_param

  Parameters passed to iterative solvers.
- struct grid2d

  Two dimensional grid data structure.
- struct Link

  Struct for Links.
- struct linked_list

  A linked list node.

Macros

- #define __FASP_HEADER__
- #define FASP_VERSION 1.8

  FASP base version information.
- #define FASP_USE_ILU ON

  For external software package support.
- #define ILU_C_VERSION ON
- #define DLMALLOC OFF
- #define NEDMALLOC OFF
- #define RS_C1 ON

  Flags for internal uses.
- #define DIAGONAL_PREF OFF
- #define SHORT short

  FASP integer and floating point numbers.
- #define INT int
- #define LONG long
- #define LONGLONG long long
- #define REAL double
- #define MAX(a, b) (((a)>(b))?(a):(b))

  Definition of max, min, abs.
- #define MIN(a, b) (((a)<(b))?(a):(b))
- #define ABS(a) (((a)>=0.0)?(a):-(a))
- #define GT(a, b) (((a)>(b))?(TRUE):(FALSE))

  Definition of >, >=, <, <=, and isnan.
- #define GE(a, b) (((a)>=(b))?(TRUE):(FALSE))
- #define LS(a, b) (((a)<(b))?(TRUE):(FALSE))
- #define LE(a, b) (((a)<=(b))?(TRUE):(FALSE))
- #define ISNAN(a) (((a)!=(a))?(TRUE):(FALSE))
- #define PUT_INT(A) printf("### DEBUG: %s = %d\n", #A, (A))

  Definition of print command in DEBUG mode.
- #define PUT_REAL(A) printf("### DEBUG: %s = %e\n", #A, (A))
- #define FASP_GSRB 1

Typedefs

- typedef struct ddenmat ddenmat
- typedef struct idenmat idenmat
- typedef struct dCSRmat dCSRmat
- typedef struct iCSRmat iCSRmat
- typedef struct dCOOmat dCOOmat
- typedef struct iCOOmat iCOOmat
- typedef struct dCSRLmat dCSRLmat
- typedef struct dSTRmat dSTRmat
- typedef struct dvector dvector
- typedef struct ivector ivector
- typedef struct grid2d grid2d
- typedef grid2d ∗ pgrid2d
- typedef const grid2d ∗ pcgrid2d
- typedef struct linked_list ListElement
- typedef ListElement ∗ LinkList

Variables

- unsigned INT total_alloc_mem
- unsigned INT total_alloc_count

  Total allocated memory amount.
- INT nx_rb
- INT ny_rb
- INT nz_rb
- INT ∗ IMAP
- INT MAXIMAP
- INT count

10.24.1 Detailed Description

Main header file for FASP.

This header file contains general constants and data structures for FASP.

Note

Only define macros and data structures, no function declarations.
Created by Chensong Zhang on 08/12/2010. Modified by Chensong Zhang on 12/13/2011. Modified by Chensong Zhang on 12/25/2011. Modified by Chensong Zhang on 01/25/2015: clean up code Modified by Chensong Zhang on 01/27/2015: remove N2C, C2N, ISTART Modified by Ludmil Zikatanov on 20151011: cosmetics.

Modified by Hongxuan Zhang on 11/28/2015: add Intel MKL PARDISO support.

10.24.2 Macro Definition Documentation

10.24.2.1 __FASP_HEADER__

```
#define __FASP_HEADER__
```

indicate fasp.h has been included before

Definition at line 36 of file fasp.h.

10.24.2.2 ABS

```
#define ABS(
            a ) (((a)>=0.0)?(a):-(a))
```

absolute value of a

Definition at line 79 of file fasp.h.

10.24.2.3 DIAGONAL_PREF

```
#define DIAGONAL_PREF OFF
```

order each row such that diagonal appears first

Definition at line 63 of file fasp.h.

10.24.2.4 DLMALLOC

```
#define DLMALLOC OFF
```

use dlmalloc instead of standard malloc

Definition at line 52 of file fasp.h.

10.24.2.5 FASP_GSRB

```
#define FASP_GSRB 1
```

MG level 0 use RedBlack Gauss Seidel Smoothing

Definition at line 1246 of file fasp.h.

10.24.2.6 FASP_USE_ILU

```
#define FASP_USE_ILU ON
```

For external software package support.

enable ILU in FASP

Definition at line 50 of file fasp.h.

10.24.2.7 FASP_VERSION

```
#define FASP_VERSION 1.8
```

FASP base version information.

faspsolver version

Definition at line 45 of file fasp.h.

10.24.2.8 GE

```
#define GE(
            a,
            b ) (((a)>=(b))?(TRUE):(FALSE))
```

is a >= b?

Definition at line 85 of file fasp.h.

10.24.2.9 GT

```
#define GT(
            a,
            b ) (((a)>(b))?(TRUE):(FALSE))
```

Definition of >, >=, <, <=, and isnan.

is a > b?

Definition at line 84 of file fasp.h.

10.24.2.10 ILU_C_VERSION

```
#define ILU_C_VERSION ON
```

use the C version of ILU functions

Definition at line 51 of file fasp.h.

10.24.2.11 INT

```
#define INT int
```

regular integer type: int or long

Definition at line 69 of file fasp.h.

10.24.2.12 ISNAN

```
#define ISNAN(
            a ) (((a)!=(a))?(TRUE):(FALSE))
```

is a == NAN?

Definition at line 88 of file fasp.h.

10.24.2.13 LE

```
#define LE(
            a,
            b ) (((a)<=(b))?(TRUE):(FALSE))
```

is a <= b?

Definition at line 87 of file fasp.h.

10.24.2.14 LONG

```
#define LONG long
```

long integer type

Definition at line 70 of file fasp.h.

10.24.2.15 LONGLONG

```
#define LONGLONG long long
```

long integer type

Definition at line 71 of file fasp.h.

10.24.2.16 LS

```
#define LS(
            a,
            b ) (((a)<(b))?(TRUE):(FALSE))
```

is a < b?

Definition at line 86 of file fasp.h.

10.24.2.17 MAX

```
#define MAX(
            a,
            b ) (((a)>(b))?(a):(b))
```

Definition of max, min, abs.

bigger one in a and b

Definition at line 77 of file fasp.h.

10.24.2.18 MIN

```
#define MIN(
              a,
              b ) (((a)<(b))?(a):(b))
```

smaller one in a and b

Definition at line 78 of file fasp.h.

10.24.2.19 NEDMALLOC

```
#define NEDMALLOC OFF
```

use nedmalloc instead of standard malloc

Definition at line 53 of file fasp.h.

10.24.2.20 PUT_INT

```
#define PUT_INT(
              A ) printf("### DEBUG: %s = %d\n", #A, (A))
```

Definition of print command in DEBUG mode.

print an integer

Definition at line 93 of file fasp.h.

10.24.2.21 PUT_REAL

```
#define PUT_REAL(
              A ) printf("### DEBUG: %s = %e\n", #A, (A))
```

print a real num

Definition at line 94 of file fasp.h.

10.24.2.22 REAL

```
#define REAL double
```

float type

Definition at line 72 of file fasp.h.

10.24.2.23 RS_C1

```
#define RS_C1 ON
```

Flags for internal uses.

Warning

Change the following marcos with caution!CF splitting of RS: check C1 Criterion

Definition at line 61 of file fasp.h.

10.24.2.24 SHORT

```
#define SHORT short
```

FASP integer and floating point numbers.

short integer type

Definition at line 68 of file fasp.h.

10.24.3 Typedef Documentation

10.24.3.1 dCOOmat

```
typedef struct dCOOmat dCOOmat
```

Sparse matrix of REAL type in COO format

10.24.3.2 dCSRLmat

```
typedef struct dCSRLmat dCSRLmat
```

Sparse matrix of REAL type in CSRL format

10.24.3.3 dCSRmat

```
typedef struct dCSRmat dCSRmat
```

Sparse matrix of REAL type in CSR format

10.24.3.4 ddenmat

```
typedef struct ddenmat ddenmat
```

Dense matrix of REAL type

10.24.3.5 dSTRmat

```
typedef struct dSTRmat dSTRmat
```

Structured matrix of REAL type

10.24.3.6 dvector

```
typedef struct dvector dvector
```

Vector of REAL type

10.24.3.7 grid2d

```
typedef struct grid2d grid2d
```

2D grid type for plotting

10.24.3.8 iCOOmat

```
typedef struct iCOOmat iCOOmat
```

Sparse matrix of INT type in COO format

10.24.3.9 iCSRmat

```
typedef struct iCSRmat iCSRmat
```

Sparse matrix of INT type in CSR format

10.24.3.10 idenmat

```
typedef struct idenmat idenmat
```

Dense matrix of INT type

10.24.3.11 ivector

```
typedef struct ivector ivector
```

Vector of INT type

10.24.3.12 LinkList

```
typedef ListElement* LinkList
```

List of linkslinked list

Definition at line 1241 of file fasp.h.

10.24.3.13 ListElement

```
typedef struct linked_list ListElement
```

Linked element in list

10.24.3.14 pcgrid2d

```
typedef const grid2d* pcgrid2d
```

Grid in 2d

Definition at line 1195 of file fasp.h.

10.24.3.15 pgrid2d

```
typedef grid2d* pgrid2d
```

Grid in 2d

Definition at line 1193 of file fasp.h.

10.24.4 Variable Documentation

10.24.4.1 count

```
INT count
```

Counter for multiple calls

10.24.4.2 IMAP

```
INT* IMAP
```

Red Black Gs Smoother imap

10.24.4.3 MAXIMAP

```
INT MAXIMAP
```

Red Black Gs Smoother max DOFs of reservoir

10.24.4.4 nx_rb

```
INT nx_rb
```

Red Black Gs Smoother Nx

10.24.4.5 ny_rb

```
INT ny_rb
```

Red Black Gs Smoother Ny

10.24.4.6 nz_rb

```
INT nz_rb
```

Red Black Gs Smoother Nz

10.24.4.7 total_alloc_count

```
unsigned INT total_alloc_count
```

Total allocated memory amount.

total allocation times

Definition at line 33 of file memory.c.

10.24.4.8 total_alloc_mem

```
unsigned INT total_alloc_mem
```

total allocated memory

Definition at line 32 of file memory.c.

10.25 fasp_block.h File Reference

Header file for FASP block matrices.

```
#include "fasp.h"
```

Data Structures

- struct dBSRmat

  Block sparse row storage matrix of REAL type.
- struct dBLCmat

  Block REAL CSR matrix format.
- struct block_iCSRmat

  Block INT CSR matrix format.
- struct block_dvector

  Block REAL vector structure.
- struct block_ivector

  Block INT vector structure.
- struct block_Reservoir

  Block REAL matrix format for reservoir simulation.
- struct block_BSR

  Block REAL matrix format for reservoir simulation.
- struct AMG_data_bsr

  Data for multigrid levels. (BSR format)
- struct precond_diagbsr

  Data passed to diagnal preconditioner for dBSRmat matrices.
- struct precond_data_bsr

  Data passed to the preconditioners.
- struct precond_block_reservoir_data

  Data passed to the preconditioner for reservoir simulation problems.
- struct precond_block_data

  Data passed to the preconditioner for block preconditioning for dBLCmat format.
- struct precond_FASP_blkoil_data

  Data passed to the preconditioner for preconditioning reservoir simulation problems.
- struct precond_sweeping_data

  Data passed to the preconditioner for sweeping preconditioning.

Macros

- #define __FASPBLOCK_HEADER__
- #define SMOOTHER_BLKOIL 11

  Definition of specialized smoother types.
- #define SMOOTHER_SPETEN 19

Typedefs

- typedef struct dBSRmat dBSRmat
- typedef struct dBLCmat dBLCmat
- typedef struct block_iCSRmat block_iCSRmat
- typedef struct block_dvector block_dvector
- typedef struct block_ivector block_ivector
- typedef struct block_Reservoir block_Reservoir
- typedef struct block_BSR block_BSR
- typedef struct precond_block_reservoir_data precond_block_reservoir_data

10.25.1 Detailed Description

Header file for FASP block matrices.

Note

This header file contains definitions of block matrices, including grid-major type and variable-major type. In this header, we only define macros and data structures, not function declarations.
Created by Chensong Zhang on 05/21/2010. Modified by Xiaozhe Hu on 05/28/2010: add precond_block_reservoir_data. Modified by Xiaozhe Hu on 06/15/2010: modify precond_block_reservoir_data. Modified by Chensong Zhang on 10/11/2010: add BSR data. Modified by Chensong Zhang on 10/17/2012: modify comments.

Modified by Ludmil Zikatanov on 20151011: cosmetics.

10.25.2 Macro Definition Documentation

10.25.2.1 __FASPBLOCK_HEADER__

```
#define __FASPBLOCK_HEADER__
```

indicate fasp_block.h has been included before

Definition at line 22 of file fasp_block.h.

10.25.2.2 SMOOTHER_BLKOIL

```
#define SMOOTHER_BLKOIL 11
```

Definition of specialized smoother types.

Used in monolithic AMG for black-oil

Definition at line 27 of file fasp_block.h.

10.25.2.3 SMOOTHER_SPETEN

```
#define SMOOTHER_SPETEN 19
```

Used in monolithic AMG for black-oil

Definition at line 28 of file fasp_block.h.

10.25.3 Typedef Documentation

10.25.3.1 block_BSR

```
typedef struct block_BSR block_BSR
```

Block of BSR matrices of REAL type

10.25.3.2 block_dvector

```
typedef struct block_dvector block_dvector
```

Vector of REAL type in Block format

10.25.3.3 block_iCSRmat

```
typedef struct block_iCSRmat block_iCSRmat
```

Matrix of INT type in Block CSR format

10.25.3.4 block_ivector

```
typedef struct block_ivector block_ivector
```

Vector of INT type in Block format

10.25.3.5 block_Reservoir

```
typedef struct block_Reservoir block_Reservoir
```

Special block matrix for Reservoir Simulation

10.25.3.6 dBLCmat

```
typedef struct dBLCmat dBLCmat
```

Matrix of REAL type in Block CSR format

10.25.3.7 dBSRmat

```
typedef struct dBSRmat dBSRmat
```

Matrix of REAL type in BSR format

10.25.3.8 precond_block_reservoir_data

```
typedef struct precond_block_reservoir_data precond_block_reservoir_data
```

Precond data for Reservoir Simulation

10.26 fasp_const.h File Reference

Definition of all kinds of messages, including error messages, solver types, etc.

Macros

- #define FASP_SUCCESS 0

  Definition of return status and error messages.
- #define ERROR_OPEN_FILE -10
- #define ERROR_WRONG_FILE -11
- #define ERROR_INPUT_PAR -13
- #define ERROR_REGRESS -14
- #define ERROR_MAT_SIZE -15
- #define ERROR_NUM_BLOCKS -18
- #define ERROR_MISC -19
- #define ERROR_ALLOC_MEM -20
- #define ERROR_DATA_STRUCTURE -21
- #define ERROR_DATA_ZERODIAG -22
- #define ERROR_DUMMY_VAR -23
- #define ERROR_AMG_INTERP_TYPE -30
- #define ERROR_AMG_SMOOTH_TYPE -31
- #define ERROR_AMG_COARSE_TYPE -32
- #define ERROR_AMG_COARSEING -33
- #define ERROR_SOLVER_TYPE -40
- #define ERROR_SOLVER_PRECTYPE -41
- #define ERROR_SOLVER_STAG -42
- #define ERROR_SOLVER_SOLSTAG -43

- #define ERROR_SOLVER_TOLSMALL -44
- #define ERROR_SOLVER_ILUSETUP -45
- #define ERROR_SOLVER_MISC -46
- #define ERROR_SOLVER_MAXIT -48
- #define ERROR_SOLVER_EXIT -49
- #define ERROR_QUAD_TYPE -60
- #define ERROR_QUAD_DIM -61
- #define ERROR_LIC_TYPE -80
- #define ERROR_UNKNOWN -99
- #define TRUE 1

    Definition of logic type.
- #define FALSE 0
- #define ON 1

    Definition of switch.
- #define OFF 0
- #define PRINT_NONE 0

    Print level for all subroutines – not including DEBUG output.
- #define PRINT_MIN 1
- #define PRINT_SOME 2
- #define PRINT_MORE 4
- #define PRINT_MOST 8
- #define PRINT_ALL 10
- #define MAT_FREE 0

    Definition of matrix format.
- #define MAT_CSR 1
- #define MAT_BSR 2
- #define MAT_STR 3
- #define MAT_BLC 4
- #define MAT_bBSR 5
- #define MAT_CSRL 6
- #define MAT_SymCSR 7
- #define SOLVER_DEFAULT 0

    Definition of solver types for iterative methods.
- #define SOLVER_CG 1
- #define SOLVER_BiCGstab 2
- #define SOLVER_VBiCGstab 9
- #define SOLVER_MinRes 3
- #define SOLVER_GMRES 4
- #define SOLVER_VGMRES 5
- #define SOLVER_VFGMRES 6
- #define SOLVER_GCG 7
- #define SOLVER_GCR 8
- #define SOLVER_SCG 11
- #define SOLVER_SBiCGstab 12
- #define SOLVER_SMinRes 13
- #define SOLVER_SGMRES 14
- #define SOLVER_SVGMRES 15
- #define SOLVER_SVFGMRES 16
- #define SOLVER_SGCG 17
- #define SOLVER_AMG 21

- #define SOLVER_FMG 22
- #define SOLVER_SUPERLU 31
- #define SOLVER_UMFPACK 32
- #define SOLVER_MUMPS 33
- #define SOLVER_PARDISO 34
- #define STOP_REL_RES 1

    Definition of iterative solver stopping criteria types.
- #define STOP_REL_PRECRES 2
- #define STOP_MOD_REL_RES 3
- #define PREC_NULL 0

    Definition of preconditioner type for iterative methods.
- #define PREC_DIAG 1
- #define PREC_AMG 2
- #define PREC_FMG 3
- #define PREC_ILU 4
- #define PREC_SCHWARZ 5
- #define ILUk 1

    Type of ILU methods.
- #define ILUt 2
- #define ILUtp 3
- #define SCHWARZ_FORWARD 1

    Type of Schwarz smoother.
- #define SCHWARZ_BACKWARD 2
- #define SCHWARZ_SYMMETRIC 3
- #define CLASSIC_AMG 1

    Definition of AMG types.
- #define SA_AMG 2
- #define UA_AMG 3
- #define PAIRWISE 1

    Definition of aggregation types.
- #define VMB 2
- #define V_CYCLE 1

    Definition of cycle types.
- #define W_CYCLE 2
- #define AMLI_CYCLE 3
- #define NL_AMLI_CYCLE 4
- #define SMOOTHER_JACOBI 1

    Definition of standard smoother types.
- #define SMOOTHER_GS 2
- #define SMOOTHER_SGS 3
- #define SMOOTHER_CG 4
- #define SMOOTHER_SOR 5
- #define SMOOTHER_SSOR 6
- #define SMOOTHER_GSOR 7
- #define SMOOTHER_SGSOR 8
- #define SMOOTHER_POLY 9
- #define SMOOTHER_L1DIAG 10
- #define COARSE_RS 1

    Definition of coarsening types.

- #define COARSE_RSP 2
- #define COARSE_CR 3
- #define COARSE_AC 4
- #define COARSE_MIS 5
- #define INTERP_DIR 1

    Definition of interpolation types.
- #define INTERP_STD 2
- #define INTERP_ENG 3
- #define G0PT -5

    Type of vertices (DOFs) for coarsening.
- #define UNPT -1
- #define FGPT 0
- #define CGPT 1
- #define ISPT 2
- #define NO_ORDER 0

    Definition of smoothing order.
- #define CF_ORDER 1
- #define ILU_MC_OMP 1
- #define USERDEFINED 0

    Type of ordering for smoothers.
- #define CPFIRST 1
- #define FPFIRST -1
- #define ASCEND 12
- #define DESCEND 21
- #define BIGREAL 1e+20

    Some global constants.
- #define SMALLREAL 1e-20
- #define SMALLREAL2 1e-40
- #define MAX_REFINE_LVL 20
- #define MAX_AMG_LVL 20
- #define MIN_CDOF 20
- #define MIN_CRATE 0.9
- #define MAX_CRATE 20.0
- #define MAX_RESTART 20
- #define MAX_STAG 20
- #define STAG_RATIO 1e-4
- #define OPENMP_HOLDS 2000

10.26.1 Detailed Description

Definition of all kinds of messages, including error messages, solver types, etc.

Note

This is internal use only. Do NOT change.
Created by Chensong Zhang on 03/20/2010. Modified by Chensong Zhang on 12/06/2011. Modified by Chensong Zhang on 12/25/2011. Modified by Chensong Zhang on 04/22/2012. Modified by Ludmil Zikatanov on 02/15/2013: CG -> SMOOTHER_CG. Modified by Chensong Zhang on 02/16/2013: GS -> SMOOTHER_GS, etc. Modified by Chensong Zhang on 04/09/2013: Add safe Krylov methods. Modified by Chensong Zhang on 09/22/2013: Clean up Doxygen.

Modified by Chensong Zhang on 09/17/2013: Filename changed from message.h.

10.26.2 Macro Definition Documentation

10.26.2.1 AMLI_CYCLE

```
#define AMLI_CYCLE 3
```

AMLI-cycle

Definition at line 178 of file fasp_const.h.

10.26.2.2 ASCEND

```
#define ASCEND 12
```

Ascending order

Definition at line 233 of file fasp_const.h.

10.26.2.3 BIGREAL

```
#define BIGREAL 1e+20
```

Some global constants.

A large real number

Definition at line 239 of file fasp_const.h.

10.26.2.4 CF_ORDER

```
#define CF_ORDER 1
```

C/F order smoothing

Definition at line 224 of file fasp_const.h.

10.26.2.5 CGPT

```
#define CGPT 1
```

Coarse grid points

Definition at line 217 of file fasp_const.h.

10.26.2.6 CLASSIC_AMG

```
#define CLASSIC_AMG 1
```

Definition of AMG types.

classic AMG

Definition at line 163 of file fasp_const.h.

10.26.2.7 COARSE_AC

```
#define COARSE_AC 4
```

Aggressive coarsening

Definition at line 201 of file fasp_const.h.

10.26.2.8 COARSE_CR

```
#define COARSE_CR 3
```

Compatible relaxation

Definition at line 200 of file fasp_const.h.

10.26.2.9 COARSE_MIS

```
#define COARSE_MIS 5
```

Aggressive coarsening based on MIS

Definition at line 202 of file fasp_const.h.

10.26.2.10 COARSE_RS

```
#define COARSE_RS 1
```

Definition of coarsening types.

Classical

Definition at line 198 of file fasp_const.h.

10.26.2.11 COARSE_RSP

```
#define COARSE_RSP 2
```

Classical, with positive offdiags

Definition at line 199 of file fasp_const.h.

10.26.2.12 CPFIRST

```
#define CPFIRST 1
```

C-points first order

Definition at line 231 of file fasp_const.h.

10.26.2.13 DESCEND

```
#define DESCEND 21
```

Descending order

Definition at line 234 of file fasp_const.h.

10.26.2.14 ERROR_ALLOC_MEM

```
#define ERROR_ALLOC_MEM -20
```

fail to allocate memory

Definition at line 37 of file fasp_const.h.

10.26.2.15 ERROR_AMG_COARSE_TYPE

```
#define ERROR_AMG_COARSE_TYPE -32
```

unknown coarsening type

Definition at line 44 of file fasp_const.h.

10.26.2.16 ERROR_AMG_COARSEING

```
#define ERROR_AMG_COARSEING -33
```

coarsening step failed to complete

Definition at line 45 of file fasp_const.h.

10.26.2.17 ERROR_AMG_INTERP_TYPE

```
#define ERROR_AMG_INTERP_TYPE -30
```

unknown interpolation type

Definition at line 42 of file fasp_const.h.

10.26.2.18 ERROR_AMG_SMOOTH_TYPE

```
#define ERROR_AMG_SMOOTH_TYPE -31
```

unknown smoother type

Definition at line 43 of file fasp_const.h.

10.26.2.19 ERROR_DATA_STRUCTURE

```
#define ERROR_DATA_STRUCTURE -21
```

problem with data structures

Definition at line 38 of file fasp_const.h.

10.26.2.20 ERROR_DATA_ZERODIAG

```
#define ERROR_DATA_ZERODIAG -22
```

matrix has zero diagonal entries

Definition at line 39 of file fasp_const.h.

10.26.2.21 ERROR_DUMMY_VAR

```
#define ERROR_DUMMY_VAR -23
```

unexpected input data

Definition at line 40 of file fasp_const.h.

10.26.2.22 ERROR_INPUT_PAR

```
#define ERROR_INPUT_PAR -13
```

wrong input argument

Definition at line 31 of file fasp_const.h.

10.26.2.23 ERROR_LIC_TYPE

```
#define ERROR_LIC_TYPE -80
```

wrong license type

Definition at line 60 of file fasp_const.h.

10.26.2.24 ERROR_MAT_SIZE

```
#define ERROR_MAT_SIZE -15
```

wrong problem size

Definition at line 33 of file fasp_const.h.

10.26.2.25 ERROR_MISC

```
#define ERROR_MISC -19
```

other error

Definition at line 35 of file fasp_const.h.

10.26.2.26 ERROR_NUM_BLOCKS

```
#define ERROR_NUM_BLOCKS -18
```

wrong number of blocks

Definition at line 34 of file fasp_const.h.

10.26.2.27 ERROR_OPEN_FILE

```
#define ERROR_OPEN_FILE -10
```

fail to open a file

Definition at line 29 of file fasp_const.h.

10.26.2.28 ERROR_QUAD_DIM

```
#define ERROR_QUAD_DIM -61
```

unsupported quadrature dim

Definition at line 58 of file fasp_const.h.

10.26.2.29 ERROR_QUAD_TYPE

```
#define ERROR_QUAD_TYPE -60
```

unknown quadrature type

Definition at line 57 of file fasp_const.h.

10.26.2.30 ERROR_REGRESS

```
#define ERROR_REGRESS -14
```

regression test fail

Definition at line 32 of file fasp_const.h.

10.26.2.31 ERROR_SOLVER_EXIT

```
#define ERROR_SOLVER_EXIT -49
```

solver does not quit successfully

Definition at line 55 of file fasp_const.h.

10.26.2.32 ERROR_SOLVER_ILUSETUP

```
#define ERROR_SOLVER_ILUSETUP -45
```

ILU setup error

Definition at line 52 of file fasp_const.h.

10.26.2.33 ERROR_SOLVER_MAXIT

```
#define ERROR_SOLVER_MAXIT -48
```

maximal iteration number exceeded

Definition at line 54 of file fasp_const.h.

10.26.2.34 ERROR_SOLVER_MISC

```
#define ERROR_SOLVER_MISC -46
```

misc solver error during run time

Definition at line 53 of file fasp_const.h.

10.26.2.35 ERROR_SOLVER_PRECTYPE

```
#define ERROR_SOLVER_PRECTYPE -41
```

unknown precond type

Definition at line 48 of file fasp_const.h.

10.26.2.36 ERROR_SOLVER_SOLSTAG

```
#define ERROR_SOLVER_SOLSTAG -43
```

solver's solution is too small

Definition at line 50 of file fasp_const.h.

10.26.2.37 ERROR_SOLVER_STAG

```
#define ERROR_SOLVER_STAG -42
```

solver stagnates

Definition at line 49 of file fasp_const.h.

10.26.2.38 ERROR_SOLVER_TOLSMALL

```
#define ERROR_SOLVER_TOLSMALL -44
```

solver's tolerance is too small

Definition at line 51 of file fasp_const.h.

10.26.2.39 ERROR_SOLVER_TYPE

```
#define ERROR_SOLVER_TYPE -40
```

unknown solver type

Definition at line 47 of file fasp_const.h.

10.26.2.40 ERROR_UNKNOWN

```
#define ERROR_UNKNOWN -99
```

an unknown error type

Definition at line 62 of file fasp_const.h.

10.26.2.41 ERROR_WRONG_FILE

```
#define ERROR_WRONG_FILE -11
```

input contains wrong format

Definition at line 30 of file fasp_const.h.

10.26.2.42 FALSE

```
#define FALSE 0
```

logic FALSE

Definition at line 68 of file fasp_const.h.

10.26.2.43 FASP_SUCCESS

```
#define FASP_SUCCESS 0
```

Definition of return status and error messages.

return from function successfully

Definition at line 27 of file fasp_const.h.

10.26.2.44 FGPT

```
#define FGPT 0
```

Fine grid points

Definition at line 216 of file fasp_const.h.

10.26.2.45 FPFIRST

```
#define FPFIRST -1
```

F-points first order

Definition at line 232 of file fasp_const.h.

10.26.2.46 G0PT

```
#define G0PT -5
```

Type of vertices (DOFs) for coarsening.

Cannot fit in aggregates

Definition at line 214 of file fasp_const.h.

10.26.2.47 ILU_MC_OMP

```
#define ILU_MC_OMP 1
```

Multi-colors Parallel smoothing

Definition at line 225 of file fasp_const.h.

10.26.2.48 ILUk

```
#define ILUk 1
```

Type of ILU methods.

ILUk

Definition at line 149 of file fasp_const.h.

10.26.2.49 ILUt

```
#define ILUt 2
```

ILUt

Definition at line 150 of file fasp_const.h.

10.26.2.50 ILUtp

```
#define ILUtp 3
```

ILUtp

Definition at line 151 of file fasp_const.h.

10.26.2.51 INTERP_DIR

```
#define INTERP_DIR 1
```

Definition of interpolation types.

Direct interpolation

Definition at line 207 of file fasp_const.h.

10.26.2.52 INTERP_ENG

```
#define INTERP_ENG 3
```

energy minimization interpolation

Definition at line 209 of file fasp_const.h.

10.26.2.53 INTERP_STD

```
#define INTERP_STD 2
```

Standard interpolation

Definition at line 208 of file fasp_const.h.

10.26.2.54 ISPT

```
#define ISPT 2
```

Isolated points

Definition at line 218 of file fasp_const.h.

10.26.2.55 MAT_bBSR

```
#define MAT_bBSR 5
```

block matrix of BSR for bordered systems

Definition at line 94 of file fasp_const.h.

10.26.2.56 MAT_BLC

```
#define MAT_BLC 4
```

block matrix of CSR

Definition at line 93 of file fasp_const.h.

10.26.2.57 MAT_BSR

```
#define MAT_BSR 2
```

block-wise compressed sparse row

Definition at line 91 of file fasp_const.h.

10.26.2.58 MAT_CSR

```
#define MAT_CSR 1
```

compressed sparse row

Definition at line 90 of file fasp_const.h.

10.26.2.59 MAT_CSRL

```
#define MAT_CSRL 6
```

modified CSR to reduce cache missing

Definition at line 95 of file fasp_const.h.

10.26.2.60 MAT_FREE

```
#define MAT_FREE 0
```

Definition of matrix format.

matrix-free format: only mxv action

Definition at line 89 of file fasp_const.h.

10.26.2.61 MAT_STR

```
#define MAT_STR 3
```

structured sparse matrix

Definition at line 92 of file fasp_const.h.

10.26.2.62 MAT_SymCSR

```
#define MAT_SymCSR 7
```

symmetric CSR format

Definition at line 96 of file fasp_const.h.

10.26.2.63 MAX_AMG_LVL

```
#define MAX_AMG_LVL 20
```

Maximal AMG coarsening level

Definition at line 243 of file fasp_const.h.

10.26.2.64 MAX_CRATE

```
#define MAX_CRATE 20.0
```

Maximal coarsening ratio

Definition at line 246 of file fasp_const.h.

10.26.2.65 MAX_REFINE_LVL

```
#define MAX_REFINE_LVL 20
```

Maximal refinement level

Definition at line 242 of file fasp_const.h.

10.26.2.66 MAX_RESTART

```
#define MAX_RESTART 20
```

Maximal restarting number

Definition at line 247 of file fasp_const.h.

10.26.2.67 MAX_STAG

```
#define MAX_STAG 20
```

Maximal number of stagnation times

Definition at line 248 of file fasp_const.h.

10.26.2.68 MIN_CDOF

```
#define MIN_CDOF 20
```

Minimal number of coarsest variables

Definition at line 244 of file fasp_const.h.

10.26.2.69 MIN_CRATE

```
#define MIN_CRATE 0.9
```

Minimal coarsening ratio

Definition at line 245 of file fasp_const.h.

10.26.2.70 NL_AMLI_CYCLE

```
#define NL_AMLI_CYCLE 4
```

Nonlinear AMLI-cycle

Definition at line 179 of file fasp_const.h.

10.26.2.71 NO_ORDER

```
#define NO_ORDER 0
```

Definition of smoothing order.

Natural order smoothing

Definition at line 223 of file fasp_const.h.

10.26.2.72 OFF

```
#define OFF 0
```

turn off certain parameter

Definition at line 74 of file fasp_const.h.

10.26.2.73 ON

```
#define ON 1
```

Definition of switch.

turn on certain parameter

Definition at line 73 of file fasp_const.h.

10.26.2.74 OPENMP_HOLDS

```
#define OPENMP_HOLDS 2000
```

Smallest size for OpenMP version

Definition at line 250 of file fasp_const.h.

10.26.2.75 PAIRWISE

```
#define PAIRWISE 1
```

Definition of aggregation types.

pairwise aggregation

Definition at line 170 of file fasp_const.h.

10.26.2.76 PREC_AMG

```
#define PREC_AMG 2
```

with AMG precond

Definition at line 141 of file fasp_const.h.

10.26.2.77 PREC_DIAG

```
#define PREC_DIAG 1
```

with diagonal precond

Definition at line 140 of file fasp_const.h.

10.26.2.78 PREC_FMG

```
#define PREC_FMG 3
```

with full AMG precond

Definition at line 142 of file fasp_const.h.

10.26.2.79 PREC_ILU

```
#define PREC_ILU 4
```

with ILU precond

Definition at line 143 of file fasp_const.h.

10.26.2.80 PREC_NULL

```
#define PREC_NULL 0
```

Definition of preconditioner type for iterative methods.

with no precond

Definition at line 139 of file fasp_const.h.

10.26.2.81 PREC_SCHWARZ

```
#define PREC_SCHWARZ 5
```

with Schwarz preconditioner

Definition at line 144 of file fasp_const.h.

10.26.2.82 PRINT_ALL

```
#define PRINT_ALL 10
```

all: all printouts, including files

Definition at line 84 of file fasp_const.h.

10.26.2.83 PRINT_MIN

```
#define PRINT_MIN 1
```

quiet: print error, important warnings

Definition at line 80 of file fasp_const.h.

10.26.2.84 PRINT_MORE

```
#define PRINT_MORE 4
```

more: print some useful debug info

Definition at line 82 of file fasp_const.h.

10.26.2.85 PRINT_MOST

```
#define PRINT_MOST 8
```

most: maximal printouts, no files

Definition at line 83 of file fasp_const.h.

10.26.2.86 PRINT_NONE

```
#define PRINT_NONE 0
```

Print level for all subroutines – not including DEBUG output.

silent: no printout at all

Definition at line 79 of file fasp_const.h.

10.26.2.87 PRINT_SOME

```
#define PRINT_SOME 2
```

some: print less important warnings

Definition at line 81 of file fasp_const.h.

10.26.2.88 SA_AMG

```
#define SA_AMG 2
```

smoothed aggregation AMG

Definition at line 164 of file fasp_const.h.

10.26.2.89 SCHWARZ_BACKWARD

```
#define SCHWARZ_BACKWARD 2
```

Backward ordering

Definition at line 157 of file fasp_const.h.

10.26.2.90 SCHWARZ_FORWARD

```
#define SCHWARZ_FORWARD 1
```

Type of Schwarz smoother.

Forward ordering

Definition at line 156 of file fasp_const.h.

10.26.2.91 SCHWARZ_SYMMETRIC

```
#define SCHWARZ_SYMMETRIC 3
```

Symmetric smoother

Definition at line 158 of file fasp_const.h.

10.26.2.92 SMALLREAL

```
#define SMALLREAL 1e-20
```

A small real number

Definition at line 240 of file fasp_const.h.

10.26.2.93 SMALLREAL2

```
#define SMALLREAL2 1e-40
```

An extremely small real number

Definition at line 241 of file fasp_const.h.

10.26.2.94 SMOOTHER_CG

```
#define SMOOTHER_CG 4
```

CG as a smoother

Definition at line 187 of file fasp_const.h.

10.26.2.95 SMOOTHER_GS

```
#define SMOOTHER_GS 2
```

Gauss-Seidel smoother

Definition at line 185 of file fasp_const.h.

10.26.2.96 SMOOTHER_GSOR

```
#define SMOOTHER_GSOR 7
```

GS + SOR smoother

Definition at line 190 of file fasp_const.h.

10.26.2.97 SMOOTHER_JACOBI

```
#define SMOOTHER_JACOBI 1
```

Definition of standard smoother types.

Jacobi smoother

Definition at line 184 of file fasp_const.h.

10.26.2.98 SMOOTHER_L1DIAG

```
#define SMOOTHER_L1DIAG 10
```

L1 norm diagonal scaling smoother

Definition at line 193 of file fasp_const.h.

10.26.2.99 SMOOTHER_POLY

```
#define SMOOTHER_POLY 9
```

Polynomial smoother

Definition at line 192 of file fasp_const.h.

10.26.2.100 SMOOTHER_SGS

```
#define SMOOTHER_SGS 3
```

Symmetric Gauss-Seidel smoother

Definition at line 186 of file fasp_const.h.

10.26.2.101 SMOOTHER_SGSOR

```
#define SMOOTHER_SGSOR 8
```

SGS + SSOR smoother

Definition at line 191 of file fasp_const.h.

10.26.2.102 SMOOTHER_SOR

```
#define SMOOTHER_SOR 5
```

SOR smoother

Definition at line 188 of file fasp_const.h.

10.26.2.103 SMOOTHER_SSOR

```
#define SMOOTHER_SSOR 6
```

SSOR smoother

Definition at line 189 of file fasp_const.h.

10.26.2.104 SOLVER_AMG

```
#define SOLVER_AMG 21
```

AMG as an iterative solver

Definition at line 121 of file fasp_const.h.

10.26.2.105 SOLVER_BiCGstab

```
#define SOLVER_BiCGstab 2
```

Bi-Conjugate Gradient Stabilized

Definition at line 104 of file fasp_const.h.

10.26.2.106 SOLVER_CG

```
#define SOLVER_CG 1
```

Conjugate Gradient

Definition at line 103 of file fasp_const.h.

10.26.2.107 SOLVER_DEFAULT

```
#define SOLVER_DEFAULT 0
```

Definition of solver types for iterative methods.

Use default solver in FASP

Definition at line 101 of file fasp_const.h.

10.26.2.108 SOLVER_FMG

```
#define SOLVER_FMG 22
```

Full AMG as an solver

Definition at line 122 of file fasp_const.h.

10.26.2.109 SOLVER_GCG

```
#define SOLVER_GCG 7
```

Generalized Conjugate Gradient

Definition at line 110 of file fasp_const.h.

10.26.2.110 SOLVER_GCR

```
#define SOLVER_GCR 8
```

Generalized Conjugate Residual

Definition at line 111 of file fasp_const.h.

10.26.2.111 SOLVER_GMRES

```
#define SOLVER_GMRES 4
```

Generalized Minimal Residual

Definition at line 107 of file fasp_const.h.

10.26.2.112 SOLVER_MinRes

```
#define SOLVER_MinRes 3
```

Minimal Residual

Definition at line 106 of file fasp_const.h.

10.26.2.113 SOLVER_MUMPS

```
#define SOLVER_MUMPS 33
```

Direct Solver: MUMPS

Definition at line 126 of file fasp_const.h.

10.26.2.114 SOLVER_PARDISO

```
#define SOLVER_PARDISO 34
```

Direct Solver: PARDISO

Definition at line 127 of file fasp_const.h.

10.26.2.115 SOLVER_SBiCGstab

```
#define SOLVER_SBiCGstab 12
```

BiCGstab with safety net

Definition at line 114 of file fasp_const.h.

10.26.2.116 SOLVER_SCG

```
#define SOLVER_SCG 11
```

Conjugate Gradient with safety net

Definition at line 113 of file fasp_const.h.

10.26.2.117 SOLVER_SGCG

```
#define SOLVER_SGCG 17
```

GCG with safety net

Definition at line 119 of file fasp_const.h.

10.26.2.118 SOLVER_SGMRES

```
#define SOLVER_SGMRES 14
```

GMRes with safety net

Definition at line 116 of file fasp_const.h.

10.26.2.119 SOLVER_SMinRes

```
#define SOLVER_SMinRes 13
```

MinRes with safety net

Definition at line 115 of file fasp_const.h.

10.26.2.120 SOLVER_SUPERLU

```
#define SOLVER_SUPERLU 31
```

Direct Solver: SuperLU

Definition at line 124 of file fasp_const.h.

10.26.2.121 SOLVER_SVFGMRES

```
#define SOLVER_SVFGMRES 16
```

Variable-restart FGMRES with safety net

Definition at line 118 of file fasp_const.h.

10.26.2.122 SOLVER_SVGMRES

```
#define SOLVER_SVGMRES 15
```

Variable-restart GMRES with safety net

Definition at line 117 of file fasp_const.h.

10.26.2.123 SOLVER_UMFPACK

```
#define SOLVER_UMFPACK 32
```

Direct Solver: UMFPack

Definition at line 125 of file fasp_const.h.

10.26.2.124 SOLVER_VBiCGstab

```
#define SOLVER_VBiCGstab 9
```

VBi-Conjugate Gradient Stabilized

Definition at line 105 of file fasp_const.h.

10.26.2.125 SOLVER_VFGMRES

```
#define SOLVER_VFGMRES 6
```

Variable Restarting Flexible GMRES

Definition at line 109 of file fasp_const.h.

10.26.2.126 SOLVER_VGMRES

```
#define SOLVER_VGMRES 5
```

Variable Restarting GMRES

Definition at line 108 of file fasp_const.h.

10.26.2.127 STAG_RATIO

```
#define STAG_RATIO 1e-4
```

Stagnation tolerance = tol∗STAGRATIO

Definition at line 249 of file fasp_const.h.

10.26.2.128 STOP_MOD_REL_RES

```
#define STOP_MOD_REL_RES 3
```

modified relative residual ||r||/||x||

Definition at line 134 of file fasp_const.h.

10.26.2.129 STOP_REL_PRECRES

```
#define STOP_REL_PRECRES 2
```

relative B-residual ||r||_B/||b||_B

Definition at line 133 of file fasp_const.h.

10.26.2.130 STOP_REL_RES

```
#define STOP_REL_RES 1
```

Definition of iterative solver stopping criteria types.

relative residual ||r||/||b||

Definition at line 132 of file fasp_const.h.

10.26.2.131 TRUE

```
#define TRUE 1
```

Definition of logic type.

logic TRUE

Definition at line 67 of file fasp_const.h.

10.26.2.132 UA_AMG

```
#define UA_AMG 3
```

unsmoothed aggregation AMG

Definition at line 165 of file fasp_const.h.

10.26.2.133 UNPT

```
#define UNPT -1
```

Undetermined points

Definition at line 215 of file fasp_const.h.

10.26.2.134 USERDEFINED

```
#define USERDEFINED 0
```

Type of ordering for smoothers.

User defined order

Definition at line 230 of file fasp_const.h.

10.26.2.135 V_CYCLE

```
#define V_CYCLE 1
```

Definition of cycle types.

V-cycle

Definition at line 176 of file fasp_const.h.

10.26.2.136 VMB

```
#define VMB 2
```

VMB aggregation

Definition at line 171 of file fasp_const.h.

10.26.2.137 W_CYCLE

```
#define W_CYCLE 2
```

W-cycle

Definition at line 177 of file fasp_const.h.

10.27 fmgcycle.c File Reference

Abstract non-recursive full multigrid cycle.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "mg_util.inl"
```

Functions

- void fasp_solver_fmgcycle (AMG_data *mgl, AMG_param *param)
  Solve Ax=b with non-recursive full multigrid K-cycle.

10.27.1 Detailed Description

Abstract non-recursive full multigrid cycle.

10.27.2 Function Documentation

10.27.2.1 fasp_solver_fmgcycle()

```
void fasp_solver_fmgcycle (
            AMG_data * mgl,
            AMG_param * param )
```

Solve Ax=b with non-recursive full multigrid K-cycle.

Parameters

mgl	Pointer to AMG data: AMG_data
param	Pointer to AMG parameters: AMG_param

Author

Chensong Zhang

Date

02/27/2011

Modified by Chensong Zhang on 06/01/2012: fix a bug when there is only one level. Modified by Chensong Zhang on 02/27/2013: update direct solvers. Modified by Zheng Li on 11/10/2014: update direct solvers. Modified by Hongxuan Zhang on 12/15/2015: update direct solvers.

Definition at line 34 of file fmgcycle.c.

10.28 formats.c File Reference

Subroutines for matrix format conversion.

```
#include "fasp.h"
#include "fasp_block.h"
#include "fasp_functs.h"
```

Functions

- SHORT fasp_format_dcoo_dcsr (dCOOmat ∗A, dCSRmat ∗B)

    Transform a REAL matrix from its IJ format to its CSR format.
- SHORT fasp_format_dcsr_dcoo (dCSRmat ∗A, dCOOmat ∗B)

    Transform a REAL matrix from its CSR format to its IJ format.
- SHORT fasp_format_dstr_dcsr (dSTRmat ∗A, dCSRmat ∗B)

    Transfer a 'dSTRmat' type matrix into a 'dCSRmat' type matrix.
- dCSRmat fasp_format_dblc_dcsr (dBLCmat ∗Ab)

    Form the whole dCSRmat A using blocks given in Ab.
- dCSRLmat ∗ fasp_format_dcsrl_dcsr (dCSRmat ∗A)

    Convert a dCSRmat into a dCSRLmat.
- dCSRmat fasp_format_dbsr_dcsr (dBSRmat ∗B)

    Transfer a 'dBSRmat' type matrix into a dCSRmat.
- dBSRmat fasp_format_dcsr_dbsr (dCSRmat ∗A, const INT nb)

    Transfer a dCSRmat type matrix into a dBSRmat.
- dBSRmat fasp_format_dstr_dbsr (dSTRmat ∗B)

    Transfer a 'dSTRmat' type matrix to a 'dBSRmat' type matrix.
- dCOOmat ∗ fasp_format_dbsr_dcoo (dBSRmat ∗B)

    Transfer a 'dBSRmat' type matrix to a 'dCOOmat' type matrix.

10.28.1 Detailed Description

Subroutines for matrix format conversion.

10.28.2 Function Documentation

10.28.2.1 fasp_format_dblc_dcsr()

```
dCSRmat fasp_format_dblc_dcsr (
            dBLCmat * Ab )
```

Form the whole dCSRmat A using blocks given in Ab.

Parameters

Ab	Pointer to dBLCmat matrix

Returns

dCSRmat matrix if succeed, NULL if fail

Author

Shiquan Zhang

Date

08/10/2010

Definition at line 292 of file formats.c.

10.28.2.2 fasp_format_dbsr_dcoo()

```
dCOOmat * fasp_format_dbsr_dcoo (
            dBSRmat * B )
```

Transfer a 'dBSRmat' type matrix to a 'dCOOmat' type matrix.

Parameters

B	Pointer to dBSRmat matrix

Returns

Pointer to dCOOmat matrix

Author

Zhiyang Zhou

Date

2010/10/26

Definition at line 943 of file formats.c.

10.28.2.3 fasp_format_dbsr_dcsr()

```
dCSRmat fasp_format_dbsr_dcsr (
            dBSRmat * B )
```

Transfer a 'dBSRmat' type matrix into a dCSRmat.

Parameters

B	Pointer to dBSRmat matrix

Returns

dCSRmat matrix

Author

Zhiyang Zhou

Date

10/23/2010

Modified by Chunsheng Feng, Xiaoqiang Yue on 05/24/2012

Note

Works for general nb (Xiaozhe)

Definition at line 495 of file formats.c.

10.28.2.4 fasp_format_dcoo_dcsr()

```
SHORT fasp_format_dcoo_dcsr (
            dCOOmat * A,
            dCSRmat * B )
```

Transform a REAL matrix from its IJ format to its CSR format.

Parameters

A	Pointer to dCOOmat matrix
B	Pointer to dCSRmat matrix

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Xuehai Huang

Date

08/10/2009

Definition at line 27 of file formats.c.

10.28.2.5 fasp_format_dcsr_dbsr()

```
dBSRmat fasp_format_dcsr_dbsr (
            dCSRmat * A,
            const INT nb )
```

Transfer a dCSRmat type matrix into a dBSRmat.

Parameters

A	Pointer to the dCSRmat type matrix
nb	size of each block

Returns

dBSRmat matrix

Author

Zheng Li

Date

03/27/2014

Note

modified by Xiaozhe Hu to avoid potential memory leakage problem

Definition at line 721 of file formats.c.

10.28.2.6 fasp_format_dcsr_dcoo()

```
SHORT fasp_format_dcsr_dcoo (
            dCSRmat * A,
            dCOOmat * B )
```

Transform a REAL matrix from its CSR format to its IJ format.

Parameters

A	Pointer to dCSRmat matrix
B	Pointer to dCOOmat matrix

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Xuehai Huang

Date

08/10/2009

Modified by Chunsheng Feng, Zheng Li

Date

10/12/2012

Definition at line 80 of file formats.c.

10.28.2.7 fasp_format_dcsrl_dcsr()

```
dCSRLmat * fasp_format_dcsrl_dcsr (
            dCSRmat * A )
```

Convert a dCSRmat into a dCSRLmat.

Parameters

A	Pointer to dCSRLmat matrix

Returns

Pointer to dCSRLmat matrix

Author

Zhiyang Zhou

Date

2011/01/07

Definition at line 361 of file formats.c.

10.28.2.8 fasp_format_dstr_dbsr()

```
dBSRmat fasp_format_dstr_dbsr (
            dSTRmat * B )
```

Transfer a 'dSTRmat' type matrix to a 'dBSRmat' type matrix.

Parameters

B	Pointer to dSTRmat matrix

Returns

dBSRmat matrix

Author

Zhiyang Zhou

Date

2010/10/26

Definition at line 839 of file formats.c.

10.28.2.9 fasp_format_dstr_dcsr()

```
SHORT fasp_format_dstr_dcsr (
            dSTRmat * A,
            dCSRmat * B )
```

Transfer a 'dSTRmat' type matrix into a 'dCSRmat' type matrix.

Parameters

A	Pointer to dSTRmat matrix
B	Pointer to dCSRmat matrix

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Zhiyang Zhou

Date

2010/04/29

Definition at line 117 of file formats.c.

10.29 givens.c File Reference

Givens transformation.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_aux_givens (const REAL beta, dCSRmat ∗H, dvector ∗y, REAL ∗tmp)

  Perform Givens rotations to compute y |beta∗e_1- H∗y|.

10.29.1 Detailed Description

Givens transformation.

10.29.2 Function Documentation

10.29.2.1 fasp_aux_givens()

```
void fasp_aux_givens (
            const REAL beta,
            dCSRmat * H,
            dvector * y,
            REAL * tmp )
```

Perform Givens rotations to compute y |beta∗e_1- H∗y|.

Parameters

beta	Norm of residual r_0
H	Upper Hessenberg dCSRmat matrix: (m+1)∗m
y	Minimizer of |beta∗e_1- H∗y|
tmp	Temporary work array

Author

Xuehai Huang

Date

10/19/2008

Definition at line 28 of file givens.c.

10.30 gmg_poisson.c File Reference

GMG method as an iterative solver for Poisson Problem.

```
#include <time.h>
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "gmg_util.inl"
```

Functions

- INT fasp_poisson_gmg_1D (REAL ∗u, REAL ∗b, const INT nx, const INT maxlevel, const REAL rtol, const SH↩ ORT prtlvl)

    Solve Ax=b of Poisson 1D equation by Geometric Multigrid Method.
- INT fasp_poisson_gmg_2D (REAL ∗u, REAL ∗b, const INT nx, const INT ny, const INT maxlevel, const REAL rtol, const SHORT prtlvl)

    Solve Ax=b of Poisson 2D equation by Geometric Multigrid Method.
- INT fasp_poisson_gmg_3D (REAL ∗u, REAL ∗b, const INT nx, const INT ny, const INT nz, const INT maxlevel, const REAL rtol, const SHORT prtlvl)

    Solve Ax=b of Poisson 3D equation by Geometric Multigrid Method.
- void fasp_poisson_fgmg_1D (REAL ∗u, REAL ∗b, const INT nx, const INT maxlevel, const REAL rtol, const SHORT prtlvl)

    Solve Ax=b of Poisson 1D equation by Geometric Multigrid Method (Full Multigrid)
- void fasp_poisson_fgmg_2D (REAL ∗u, REAL ∗b, const INT nx, const INT ny, const INT maxlevel, const REAL rtol, const SHORT prtlvl)

    Solve Ax=b of Poisson 2D equation by Geometric Multigrid Method (Full Multigrid)

- void fasp_poisson_fgmg_3D (REAL ∗u, REAL ∗b, const INT nx, const INT ny, const INT nz, const INT maxlevel, const REAL rtol, const SHORT prtlvl)

    Solve Ax=b of Poisson 3D equation by Geometric Multigrid Method (Full Multigrid)
- INT fasp_poisson_pcg_gmg_1D (REAL ∗u, REAL ∗b, const INT nx, const INT maxlevel, const REAL rtol, const SHORT prtlvl)

    Solve Ax=b of Poisson 1D equation by Geometric Multigrid Method (GMG preconditioned Conjugate Gradient method)
- INT fasp_poisson_pcg_gmg_2D (REAL ∗u, REAL ∗b, const INT nx, const INT ny, const INT maxlevel, const REAL rtol, const SHORT prtlvl)

    Solve Ax=b of Poisson 2D equation by Geometric Multigrid Method (GMG preconditioned Conjugate Gradient method)
- INT fasp_poisson_pcg_gmg_3D (REAL ∗u, REAL ∗b, const INT nx, const INT ny, const INT nz, const INT maxlevel, const REAL rtol, const SHORT prtlvl)

    Solve Ax=b of Poisson 3D equation by Geometric Multigrid Method (GMG preconditioned Conjugate Gradient method)

10.30.1 Detailed Description

GMG method as an iterative solver for Poisson Problem.

10.30.2 Function Documentation

10.30.2.1 fasp_poisson_fgmg_1D()

```
void fasp_poisson_fgmg_1D (
            REAL * u,
            REAL * b,
            const INT nx,
            const INT maxlevel,
            const REAL rtol,
            const SHORT prtlvl )
```

Solve Ax=b of Poisson 1D equation by Geometric Multigrid Method (Full Multigrid)

Parameters

u	Pointer to the vector of dofs
b	Pointer to the vector of right hand side
nx	Number of grids in x direction
maxlevel	Maximum levels of the multigrid
rtol	Relative tolerance to judge convergence
prtlvl	Print level for output

Author

Ziteng Wang

Date

06/07/2013

Definition at line 431 of file gmg_poisson.c.

10.30.2.2 fasp_poisson_fgmg_2D()

```
void fasp_poisson_fgmg_2D (
            REAL * u,
            REAL * b,
            const INT nx,
            const INT ny,
            const INT maxlevel,
            const REAL rtol,
            const SHORT prtlvl )
```

Solve Ax=b of Poisson 2D equation by Geometric Multigrid Method (Full Multigrid)

Parameters

u	Pointer to the vector of dofs
b	Pointer to the vector of right hand side
nx	Number of grids in x direction
ny	Number of grids in Y direction
maxlevel	Maximum levels of the multigrid
rtol	Relative tolerance to judge convergence
prtlvl	Print level for output

Author

Ziteng Wang

Date

06/07/2013

Definition at line 524 of file gmg_poisson.c.

10.30.2.3 fasp_poisson_fgmg_3D()

```
void fasp_poisson_fgmg_3D (
            REAL * u,
            REAL * b,
            const INT nx,
            const INT ny,
```

```
const INT nz,
const INT maxlevel,
const REAL rtol,
const SHORT prtlvl )
```

Solve Ax=b of Poisson 3D equation by Geometric Multigrid Method (Full Multigrid)

Parameters

u	Pointer to the vector of dofs
b	Pointer to the vector of right hand side
nx	Number of grids in x direction
ny	NUmber of grids in y direction
nz	NUmber of grids in z direction
maxlevel	Maximum levels of the multigrid
rtol	Relative tolerance to judge convergence
prtlvl	Print level for output

Author

Ziteng Wang

Date

06/07/2013

Definition at line 632 of file gmg_poisson.c.

10.30.2.4 fasp_poisson_gmg_1D()

```
INT fasp_poisson_gmg_1D (
            REAL * u,
            REAL * b,
            const INT nx,
            const INT maxlevel,
            const REAL rtol,
            const SHORT prtlvl )
```

Solve Ax=b of Poisson 1D equation by Geometric Multigrid Method.

Parameters

u	Pointer to the vector of dofs
b	Pointer to the vector of right hand side
nx	Number of grids in x direction
maxlevel	Maximum levels of the multigrid
rtol	Relative tolerance to judge convergence
prtlvl	Print level for output

Returns

Iteration number if converges; ERROR otherwise.

Author

Ziteng Wang

Date

06/07/2013

Definition at line 36 of file gmg_poisson.c.

10.30.2.5 fasp_poisson_gmg_2D()

```
INT fasp_poisson_gmg_2D (
            REAL * u,
            REAL * b,
            const INT nx,
            const INT ny,
            const INT maxlevel,
            const REAL rtol,
            const SHORT prtlvl )
```

Solve Ax=b of Poisson 2D equation by Geometric Multigrid Method.

Parameters

u	Pointer to the vector of dofs
b	Pointer to the vector of right hand side
nx	Number of grids in x direction
ny	Number of grids in y direction
maxlevel	Maximum levels of the multigrid
rtol	Relative tolerance to judge convergence
prtlvl	Print level for output

Returns

Iteration number if converges; ERROR otherwise.

Author

Ziteng Wang

Date

06/07/2013

Definition at line 160 of file gmg_poisson.c.

10.30.2.6 fasp_poisson_gmg_3D()

```
INT fasp_poisson_gmg_3D (
            REAL * u,
            REAL * b,
            const INT nx,
            const INT ny,
            const INT nz,
            const INT maxlevel,
            const REAL rtol,
            const SHORT prtlvl )
```

Solve Ax=b of Poisson 3D equation by Geometric Multigrid Method.

Parameters

u	Pointer to the vector of dofs
b	Pointer to the vector of right hand side
nx	Number of grids in x direction
ny	Number of grids in y direction
nz	Number of grids in z direction
maxlevel	Maximum levels of the multigrid
rtol	Relative tolerance to judge convergence
prtlvl	Print level for output

Returns

Iteration number if converges; ERROR otherwise.

Author

Ziteng Wang

Date

06/07/2013

Definition at line 296 of file gmg_poisson.c.

10.30.2.7 fasp_poisson_pcg_gmg_1D()

```
INT fasp_poisson_pcg_gmg_1D (
            REAL * u,
            REAL * b,
            const INT nx,
            const INT maxlevel,
            const REAL rtol,
            const SHORT prtlvl )
```

Solve Ax=b of Poisson 1D equation by Geometric Multigrid Method (GMG preconditioned Conjugate Gradient method)

Parameters

u	Pointer to the vector of dofs
b	Pointer to the vector of right hand side
nx	Number of grids in x direction
maxlevel	Maximum levels of the multigrid
rtol	Relative tolerance to judge convergence
prtlvl	Print level for output

Returns

Iteration number if converges; ERROR otherwise.

Author

Ziteng Wang

Date

06/07/2013

Definition at line 741 of file gmg_poisson.c.

10.30.2.8 fasp_poisson_pcg_gmg_2D()

```
INT fasp_poisson_pcg_gmg_2D (
            REAL * u,
            REAL * b,
            const INT nx,
            const INT ny,
            const INT maxlevel,
            const REAL rtol,
            const SHORT prtlvl )
```

Solve Ax=b of Poisson 2D equation by Geometric Multigrid Method (GMG preconditioned Conjugate Gradient method)

Parameters

u	Pointer to the vector of dofs
b	Pointer to the vector of right hand side
nx	Number of grids in x direction
ny	Number of grids in y direction
maxlevel	Maximum levels of the multigrid
rtol	Relative tolerance to judge convergence
prtlvl	Print level for output

Returns

Iteration number if converges; ERROR otherwise.

Author

Ziteng Wang

Date

06/07/2013

Definition at line 835 of file gmg_poisson.c.

10.30.2.9 fasp_poisson_pcg_gmg_3D()

```
INT fasp_poisson_pcg_gmg_3D (
            REAL * u,
            REAL * b,
            const INT nx,
            const INT ny,
            const INT nz,
            const INT maxlevel,
            const REAL rtol,
            const SHORT prtlvl )
```

Solve Ax=b of Poisson 3D equation by Geometric Multigrid Method (GMG preconditioned Conjugate Gradient method)

Parameters

u	Pointer to the vector of dofs
b	Pointer to the vector of right hand side
nx	Number of grids in x direction
ny	Number of grids in y direction
nz	Number of grids in z direction
maxlevel	Maximum levels of the multigrid
rtol	Relative tolerance to judge convergence
prtlvl	Print level for output

Returns

Iteration number if converges; ERROR otherwise.

Author

Ziteng Wang

Date

06/07/2013

Definition at line 944 of file gmg_poisson.c.

10.31 graphics.c File Reference

Subroutines for graphical output.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_dcsr_subplot (const dCSRmat *A, const char *filename, INT size)

  Write sparse matrix pattern in BMP file format.
- void fasp_dbsr_subplot (const dBSRmat *A, const char *filename, INT size)

  Write sparse matrix pattern in BMP file format.
- void fasp_grid2d_plot (pgrid2d pg, INT level)

  Output grid to a EPS file.
- INT fasp_dbsr_plot (const dBSRmat *A, const char *fname)

  Write dBSR sparse matrix pattern in BMP file format.
- INT fasp_dcsr_plot (const dCSRmat *A, const char *fname)

  Write dCSR sparse matrix pattern in BMP file format.

10.31.1 Detailed Description

Subroutines for graphical output.

10.31.2 Function Documentation

10.31.2.1 fasp_dbsr_plot()

```
void fasp_dbsr_plot (
            const dBSRmat * A,
            const char * filename )
```

Write dBSR sparse matrix pattern in BMP file format.

Parameters

A	Pointer to the dBSRmat matrix
filename	File name

Author

Chunsheng Feng

Date

11/16/2013

Note

The routine fasp_dbsr_plot writes pattern of the specified dBSRmat matrix in uncompressed BMP file format (Windows bitmap) to a binary file whose name is specified by the character string filename.

Each pixel corresponds to one matrix element. The pixel colors have the following meaning:

White structurally zero element Black zero element Blue positive element Red negative element Brown nearly zero element

Definition at line 470 of file graphics.c.

10.31.2.2 fasp_dbsr_subplot()

```
void fasp_dbsr_subplot (
            const dBSRmat * A,
            const char * filename,
            INT size )
```

Write sparse matrix pattern in BMP file format.

Parameters

A	Pointer to the dBSRmat matrix
filename	File name
size	size∗size is the picture size for the picture

Author

Chunsheng Feng

Date

11/16/2013

Note

The routine fasp_dbsr_subplot writes pattern of the specified dBSRmat matrix in uncompressed BMP file format (Windows bitmap) to a binary file whose name is specified by the character string filename.

Each pixel corresponds to one matrix element. The pixel colors have the following meaning:

White structurally zero element Black zero element Blue positive element Red negative element Brown nearly zero element

Definition at line 105 of file graphics.c.

10.31.2.3 fasp_dcsr_plot()

```
INT fasp_dcsr_plot (
            const dCSRmat * A,
            const char * fname )
```

Write dCSR sparse matrix pattern in BMP file format.

Parameters

A	Pointer to the dBSRmat matrix
fname	File name to plot to

Author

Chunsheng Feng

Date

11/16/2013

Note

The routine fasp_dcsr_plot writes pattern of the specified dCSRmat matrix in uncompressed BMP file format (Windows bitmap) to a binary file whose name is specified by the character string filename.

Each pixel corresponds to one matrix element. The pixel colors have the following meaning:

White structurally zero element Black zero element Blue positive element Red negative element Brown nearly zero element

Definition at line 630 of file graphics.c.

10.31.2.4 fasp_dcsr_subplot()

```
void fasp_dcsr_subplot (
            const dCSRmat * A,
            const char * filename,
            INT size )
```

Write sparse matrix pattern in BMP file format.

Parameters

A	Pointer to the dCSRmat matrix
filename	File name
size	size*size is the picture size for the picture

Author

Chensong Zhang

Date

03/29/2009

Note

The routine fasp_dcsr_subplot writes pattern of the specified dCSRmat matrix in uncompressed BMP file format (Windows bitmap) to a binary file whose name is specified by the character string filename.

Each pixel corresponds to one matrix element. The pixel colors have the following meaning:

White structurally zero element Blue positive element Red negative element Brown nearly zero element

Definition at line 44 of file graphics.c.

10.31.2.5 fasp_grid2d_plot()

```
void fasp_grid2d_plot (
            pgrid2d pg,
            INT level )
```

Output grid to a EPS file.

Parameters

pg	Pointer to grid in 2d
level	Number of levels

Author

Chensong Zhang

Date

03/29/2009

Definition at line 172 of file graphics.c.

10.32 ilu.c File Reference

Incomplete LU decomposition: ILUk, ILUt, ILUtp.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_qsplit (REAL ∗a, INT ∗ind, INT n, INT ncut)

  Get a quick-sort split of a real array.
- void fasp_iluk (INT n, REAL ∗a, INT ∗ja, INT ∗ia, INT lfil, REAL ∗alu, INT ∗jlu, INT iwk, INT ∗ierr, INT ∗nzlu)

  Get ILU factorization with level of fill-in k (ilu(k)) for a CSR matrix A.
- void fasp_ilut (INT n, REAL ∗a, INT ∗ja, INT ∗ia, INT lfil, REAL droptol, REAL ∗alu, INT ∗jlu, INT iwk, INT ∗ierr, INT ∗nz)

  Get incomplete LU factorization with dual truncations of a CSR matrix A.
- void fasp_ilutp (INT n, REAL ∗a, INT ∗ja, INT ∗ia, INT lfil, REAL droptol, REAL permtol, INT mbloc, REAL ∗alu, INT ∗jlu, INT iwk, INT ∗ierr, INT ∗nz)

  Get incomplete LU factorization with pivoting dual truncations of a CSR matrix A.
- void fasp_srtr (INT num, INT ∗q)

  Shell sort with hardwired increments.
- void fasp_symbfactor (INT n, INT ∗colind, INT ∗rwptr, INT levfill, INT nzmax, INT ∗nzlu, INT ∗ijlu, INT ∗uptr, INT ∗ierr)

  Symbolic factorization of a CSR matrix A in compressed sparse row format, with resulting factors stored in a single MSR data structure.

10.32.1 Detailed Description

Incomplete LU decomposition: ILUk, ILUt, ILUtp.

Note

This is a translation from SPARSEKIT Fortran version

Translated by Chunsheng Feng, 09/03/2016

10.32.2 Function Documentation

10.32.2.1 fasp_iluk()

```
void fasp_iluk (
            INT n,
            REAL * a,
            INT * ja,
            INT * ia,
            INT lfil,
            REAL * alu,
            INT * jlu,
            INT iwk,
            INT * ierr,
            INT * nzlu )
```

Get ILU factorization with level of fill-in k (ilu(k)) for a CSR matrix A.

Parameters

n	row number of A
a	nonzero entries of A
ja	integer array of column for A
ia	integer array of row pointers for A
lfil	integer. The fill-in parameter. Each row of L and each row of U will have a maximum of lfil elements (excluding the diagonal element). lfil must be .ge. 0.
alu,jlu	matrix stored in Modified Sparse Row (MSR) format containing the L and U factors together. The diagonal (stored in alu(1:n)) is inverted. Each i-th row of the alu,jlu matrix contains the i-th row of L (excluding the diagonal entry=1) followed by the i-th row of U.
jlu	integer array of length n containing the pointers to the beginning of each row of U in the matrix alu,jlu.
iwk	integer. The minimum length of arrays alu, jlu, and levs.
ierr	integer pointer. Return error message with the following meaning. 0 –> successful return. >0 –> zero pivot encountered at step number ierr. -1 –> Error. input matrix may be wrong. (The elimination process has generated a row in L or U whose length is .gt. n.) -2 –> The matrix L overflows the array al. -3 –> The matrix U overflows the array alu. -4 –> Illegal value for lfil. -5 –> zero row encountered.
nzlu	integer pointer. Return number of nonzero entries for alu and jlu

Note

: All the diagonal elements of the input matrix must be nonzero.

Author

Chunsheng Feng

Date

09/06/2016

Definition at line 136 of file ilu.c.

10.32.2.2 fasp_ilut()

```
void fasp_ilut (
            INT n,
            REAL * a,
            INT * ja,
            INT * ia,
            INT lfil,
            REAL droptol,
            REAL * alu,
            INT * jlu,
            INT iwk,
            INT * ierr,
            INT * nz )
```

Get incomplete LU factorization with dual truncations of a CSR matrix A.

Parameters

n	row number of A
a	nonzero entries of A
ja	integer array of column for A
ia	integer array of row pointers for A
lfil	integer. The fill-in parameter. Each row of L and each row of U will have a maximum of lfil elements (excluding the diagonal element). lfil must be .ge. 0.
droptol	real∗8. Sets the threshold for dropping small terms in the factorization. See below for details on dropping strategy.
alu,jlu	matrix stored in Modified Sparse Row (MSR) format containing the L and U factors together. The diagonal (stored in alu(1:n)) is inverted. Each i-th row of the alu,jlu matrix contains the i-th row of L (excluding the diagonal entry=1) followed by the i-th row of U.
iwk	integer. The lengths of arrays alu and jlu. If the arrays are not big enough to store the ILU factorizations, ilut will stop with an error message.
ierr	integer pointer. Return error message with the following meaning. 0 –> successful return. >0 –> zero pivot encountered at step number ierr. -1 –> Error. input matrix may be wrong. (The elimination process has generated a row in L or U whose length is .gt. n.) -2 –> The matrix L overflows the array al. -3 –> The matrix U overflows the array alu. -4 –> Illegal value for lfil. -5 –> zero row encountered.
nz	integer pointer. Return number of nonzero entries for alu and jlu

Note

All the diagonal elements of the input matrix must be nonzero.

Author

Chunsheng Feng

Date

09/06/2016

Definition at line 528 of file ilu.c.

10.32.2.3 fasp_ilutp()

```
void fasp_ilutp (
            INT n,
            REAL * a,
            INT * ja,
            INT * ia,
            INT lfil,
            REAL droptol,
            REAL permtol,
            INT mbloc,
            REAL * alu,
            INT * jlu,
            INT iwk,
            INT * ierr,
            INT * nz )
```

Get incomplete LU factorization with pivoting dual truncations of a CSR matrix A.

Parameters

n	row number of A
a	nonzero entries of A
ja	integer array of column for A
ia	integer array of row pointers for A
lfil	integer. The fill-in parameter. Each row of L and each row of U will have a maximum of lfil elements (excluding the diagonal element). lfil must be .ge. 0.
droptol	real∗8. Sets the threshold for dropping small terms in the factorization. See below for details on dropping strategy.
permtol	tolerance ratio used to determne whether or not to permute two columns. At step i columns i and j are permuted when abs(a(i,j))∗permtol .gt. abs(a(i,i)) [0 –> never permute; good values 0.1 to 0.01]
mbloc	integer.If desired, permuting can be done only within the diagonal blocks of size mbloc. Useful for PDE problems with several degrees of freedom.. If feature not wanted take mbloc=n.
alu,jlu	matrix stored in Modified Sparse Row (MSR) format containing the L and U factors together. The diagonal (stored in alu(1:n)) is inverted. Each i-th row of the alu,jlu matrix contains the i-th row of L (excluding the diagonal entry=1) followed by the i-th row of U.
iwk	integer. The lengths of arrays alu and jlu. If the arrays are not big enough to store the ILU factorizations, ilut will stop with an error message.
ierr	integer pointer. Return error message with the following meaning. 0 –> successful return. >0 –> zero pivot encountered at step number ierr. -1 –> Error. input matrix may be wrong. (The elimination process has generated a row in L or U whose length is .gt. n.) -2 –> The matrix L overflows the array al. -3 –> The matrix U overflows the array alu. -4 –> Illegal value for lfil. -5 –> zero row encountered.
nz	integer pointer. Return number of nonzero entries for alu and jlu

Note

: All the diagonal elements of the input matrix must be nonzero.

Author

Chunsheng Feng

Date

09/06/2016

Definition at line 963 of file ilu.c.

10.32.2.4 fasp_qsplit()

```
void void fasp_qsplit (
            REAL * a,
            INT * ind,
            INT n,
            INT ncut )
```

Get a quick-sort split of a real array.

Parameters

a	a real array. on output a(1:n) is permuted such that its elements satisfy: abs(a(i)) .ge. abs(a(ncut)) for i .lt. ncut and abs(a(i)) .le. abs(a(ncut)) for i .gt. ncut.
ind	is an integer array which permuted in the same way as a(∗).
n	size of array a.
ncut	integer.

Author

Chunsheng Feng

Date

09/06/2016

Definition at line 31 of file ilu.c.

10.32.2.5 fasp_srtr()

```
void fasp_srtr (
            INT num,
            INT * q )
```

Shell sort with hardwired increments.

Parameters

num	size of q
q	integer array.

Author

Chunsheng Feng

Date

09/06/2016

Implement shell sort, with hardwired increments. The algorithm for

sorting entries in A(0:n-1) is as follows:

inc = initialinc(n) while inc >= 1 for i = inc to n-1 j = i x = A(i) while j >= inc and A(j-inc) > x A(j) = A(j-inc) j = j-inc end while A(j) = x end for inc = nextinc(inc,n)

end while

The increments here are 1, 4, 13, 40, 121, ..., (3∗∗i - 1)/2, ... In this case, nextinc(inc,n) = (inc-1)/3. Usually shellsort would have the largest increment the largest integer of the form (3∗∗i - 1)/2 that is less than n, but here it is fixed at 121 because most sparse matrices have 121 or fewer nonzero entries per row. If this routine is expanded for a complete sparse factorization routine, or if a large number of levels of fill is allowed, then possibly it should be replaced with more efficient sorting.

Any set of increments with 1 as the first one will result in a true sorting algorithm.

Definition at line 1415 of file ilu.c.

10.32.2.6 fasp_symbfactor()

```
void fasp_symbfactor (
            INT n,
            INT * colind,
            INT * rwptr,
            INT levfill,
            INT nzmax,
            INT * nzlu,
            INT * ijlu,
            INT * uptr,
            INT * ierr )
```

Symbolic factorization of a CSR matrix A in compressed sparse row format, with resulting factors stored in a single MSR data structure.

Parameters

n	row number of A
colind	integer array of column for A
rwptr	integer array of row pointers for A
levfill	integer. Level of fill-in allowed
nzmax	integer. The maximum number of nonzero entries in the approximate factorization of a. This is the amount of storage allocated for ijlu.
nzlu	integer pointer. Return number of nonzero entries for alu and jlu
ijlu	integer array of length nzlu containing pointers to delimit rows and specify column number for stored elements of the approximate factors of A. the L and U factors are stored as one matrix.
uptr	integer array of length n containing the pointers to upper trig matrix
ierr	integer pointer. Return error message with the following meaning. 0 –> successful return. 1 –> not enough storage; check mneed.

Author

Chunsheng Feng

Date

09/06/2016

Symbolic factorization of a matrix in compressed sparse row format, ∗ with resulting factors stored in a single MSR data structure. ∗

This routine uses the CSR data structure of A in two integer vectors ∗ colind, rwptr to set up the data structure for the ILU(levfill) ∗ factorization of A in the integer vectors ijlu and uptr. Both L ∗ and U are stored in the same structure, and uptr(i) is the pointer ∗ to the beginning of the i-th row of U in ijlu. ∗

Method Used ∗ =========== ∗

The implementation assumes that the diagonal entries are ∗ nonzero, and remain nonzero throughout the elimination ∗ process. The algorithm proceeds row by row. When computing ∗ the sparsity pattern of the i-th row, the effect of row ∗ operations from previous rows is considered. Only those ∗ preceding rows j for which (i,j) is nonzero need be considered, ∗ since otherwise we would not have formed a linear combination ∗ of rows i and j. ∗

The method used has some variations possible. The definition ∗ of ILU(s) is not well specified enough to get a factorization ∗ that is uniquely defined, even in the sparsity pattern that ∗ results. For s = 0 or 1, there is not much variation, but for ∗ higher levels of fill the problem is as follows: Suppose ∗ during the decomposition while computing the nonzero pattern ∗ for row i the following principal submatrix is obtained: ∗ _______________________ ∗ | | | ∗ | | | ∗ | j,j | j,k | ∗ | | | ∗ |___________|____________| ∗ | | | ∗ | | | ∗ | i,j | i,k | ∗ | | | ∗ |___________|____________| ∗

Furthermore, suppose that entry (i,j) resulted from an earlier ∗ fill-in and has level s1, and (j,k) resulted from an earlier ∗ fill-in and has level s2: ∗ _______________________ ∗ | | | ∗ | | | ∗ | level 0 | level s2 | ∗ | | | ∗ |___________|____________| ∗ | | | ∗ | | | ∗ | level s1 | | ∗ | | | ∗ |___________|____________| ∗

When using A(j,j) to annihilate A(i,j), fill-in will be incurred ∗ in A(i,k). How should its level be defined? It would not be ∗ operated on if A(i,j) or A(j,m) had not been filled in. The ∗ version used here is to define its level as s1 + s2 +

1. However, ∗ other reasonable choices would have been min(s1,s2) or max(s1,s2). ∗ Using the sum gives a more conservative strategy in terms of the ∗ growth of the number of nonzeros as s increases. ∗

levels(n+2:nzlu) stores the levels from previous rows, ∗ that is, the s2's above. levels(1:n) stores the fill-levels ∗ of the current row (row i), which are the s1's above. ∗ levels(n+1) is not used, so levels is conformant with MSR format. ∗

Vectors used: ∗ ============= ∗

lastcol(n): ∗ The integer lastcol(k) is the row index of the last row ∗ to have a nonzero in column k, including the current ∗ row, and fill-in up to this point. So for the matrix ∗

|-----------------------—| ∗ | 11 12 15 | ∗ | 21 22 26| ∗ | 32 33 34 | ∗ | 41 43 44 | ∗ | 52 54 55 56| ∗ | 62 66| ∗ ------------------------— ∗

after step 1, lastcol() = [1 0 0 0 1 0] ∗ after step 2, lastcol() = [2 2 0 0 2 2] ∗ after step 3, lastcol() = [2 3 3 3 2 3] ∗ after step 4, lastcol() = [4 3 4 4 4 3] ∗ after step 5, lastcol() = [4 5 4 5 5 5] ∗ after step 6, lastcol() = [4 6 4 5 5 6] ∗

Note that on step 2, lastcol(5) = 2 because there is a ∗ fillin position (2,5) in the matrix. lastcol() is used ∗ to determine if a nonzero occurs in column j because ∗ it is a nonzero in the original matrix, or was a fill. ∗

rowll(n): ∗ The integer vector rowll is used to keep a linked list of ∗ the nonzeros in the current row, allowing fill-in to be ∗ introduced sensibly. rowll is initialized with the ∗ original nonzeros of the current row, and then sorted ∗ using a shell sort. A pointer called head ∗ (what ingenuity) is initialized. Note that at any ∗ point rowll may contain garbage left over from previous ∗ rows, which the linked list structure skips over. ∗ For row 4 of the matrix above, first rowll is set to ∗ rowll() = [3 1 2 5 - -], where - indicates any integer. ∗ Then the vector is sorted, which yields ∗ rowll() = [1 2 3 5 - -]. The vector is then expanded ∗ to linked list form by setting head = 1 and ∗ rowll() = [2 3 5 - 7 -], where 7 indicates termination. ∗

ijlu(nzlu): ∗ The returned nonzero structure for the LU factors. ∗ This is built up row by row in MSR format, with both L ∗ and U stored in the data structure. Another vector, uptr(n), ∗ is used to give pointers to the beginning of the upper ∗ triangular part of the LU factors in ijlu. ∗

levels(n+2:nzlu): ∗ This vector stores the fill level for each entry from ∗ all the previous rows, used to compute if the current entry ∗ will exceed the allowed levels of fill. The value in ∗ levels(m) is added to the level of fill for the element in ∗ the current row that is being reduced, to figure if ∗ a column entry is to be accepted as fill, or rejected. ∗ See the method explanation above. ∗

levels(1:n): ∗ This vector stores the fill level number for the current ∗ row's entries. If they were created as fill elements ∗ themselves, this number is added to the corresponding ∗ entry in levels(n+2:nzlu) to see if a particular column ∗ entry will ∗ be created as new fill or not. NOTE: in practice, the ∗ value in levels(1:n) is one larger than the "fill" level of ∗ the corresponding row entry, except for the diagonal ∗ entry. That is why the accept/reject test in the code ∗ is "if (levels(j) + levels(m) .le. levfill + 1)". ∗

on entry:

n = The order of the matrix A. ija = Integer array. Matrix A stored in modified sparse row format. levfill = Integer. Level of fill-in allowed. nzmax = Integer. The maximum number of nonzero entries in the approximate factorization of a. This is the amount of storage allocated for ijlu.

on return:

nzlu = The actual number of entries in the approximate factors, plus one. ijlu = Integer array of length nzlu containing pointers to delimit rows and specify column number for stored elements of the approximate factors of a. the l and u factors are stored as one matrix. uptr = Integer array of length n containing the pointers to upper trig matrix

ierr is an error flag: ierr = -i –> near zero pivot in step i ierr = 0 –> all's OK ierr = 1 –> not enough storage; check mneed. ierr = 2 –> illegal parameter

mneed = contains the actual number of elements in ldu, or the amount of additional storage needed for ldu

work arrays:

lastcol = integer array of length n containing last update of the corresponding column. levels = integer array of length n containing the level of fill-in in current row in its first n entries, and level of fill of previous rows of U in remaining part. rowll = integer array of length n containing pointers to implement a linked list for the fill-in elements.

external functions:

ifix, float, min0, srtr

Definition at line 1528 of file ilu.c.

10.33 ilu_setup_bsr.c File Reference

Setup incomplete LU decomposition for dBSRmat matrices.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void **symbfactor_** (const INT ∗n, INT ∗colind, INT ∗rwptr, const INT ∗levfill, const INT ∗nzmax, INT ∗nzlu, INT ∗ijlu, INT ∗uptr, INT ∗ierr)
- SHORT fasp_ilu_dbsr_setup (dBSRmat ∗A, ILU_data ∗iludata, ILU_param ∗iluparam)

    Get ILU decoposition of a BSR matrix A.
- SHORT fasp_ilu_dbsr_setup_levsch_omp (dBSRmat ∗A, ILU_data ∗iludata, ILU_param ∗iluparam)

    Get ILU decoposition of a BSR matrix A based on level schedule strategy.
- SHORT fasp_ilu_dbsr_setup_omp (dBSRmat ∗A, ILU_data ∗iludata, ILU_param ∗iluparam)

    Multi-threads parallel ILU decoposition of a BSR matrix A based on graph coloring.
- SHORT fasp_ilu_dbsr_setup_mc_omp (dBSRmat ∗A, dCSRmat ∗Ap, ILU_data ∗iludata, ILU_param ∗iluparam)

    Multi-threads parallel ILU decoposition of a BSR matrix A based on graph coloring.

10.33.1 Detailed Description

Setup incomplete LU decomposition for dBSRmat matrices.

10.33.2 Function Documentation

10.33.2.1 fasp_ilu_dbsr_setup()

```
SHORT fasp_ilu_dbsr_setup (
            dBSRmat * A,
            ILU_data * iludata,
            ILU_param * iluparam )
```

Get ILU decoposition of a BSR matrix A.

Parameters

A	Pointer to dBSRmat matrix
iludata	Pointer to ILU_data
iluparam	Pointer to ILU_param

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Shiquan Zhang, Xiaozhe Hu

Date

11/08/2010

Note

Works for general nb (Xiaozhe)
Change the size of work space by Zheng Li 04/26/2015.

Definition at line 45 of file ilu_setup_bsr.c.

10.33.2.2 fasp_ilu_dbsr_setup_levsch_omp()

```
SHORT fasp_ilu_dbsr_setup_levsch_omp (
            dBSRmat * A,
            ILU_data * iludata,
            ILU_param * iluparam )
```

Get ILU decoposition of a BSR matrix A based on level schedule strategy.

Parameters

A	Pointer to dBSRmat matrix
iludata	Pointer to ILU_data
iluparam	Pointer to ILU_param

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Zheng Li

Date

12/04/2016

Note

Only works for 1, 2, 3 nb (Zheng)

Definition at line 850 of file ilu_setup_bsr.c.

10.33.2.3 fasp_ilu_dbsr_setup_mc_omp()

```
SHORT fasp_ilu_dbsr_setup_mc_omp (
            dBSRmat * A,
            dCSRmat * Ap,
            ILU_data * iludata,
            ILU_param * iluparam )
```

Multi-threads parallel ILU decoposition of a BSR matrix A based on graph coloring.

Parameters

A	Pointer to dBSRmat matrix
Ap	Pointer to dCSRmat matrix and provide sparsity pattern
iludata	Pointer to ILU_data
iluparam	Pointer to ILU_param

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Zheng Li

Date

12/04/2016

Note

Only works for 1, 2, 3 nb (Zheng)

Definition at line 1084 of file ilu_setup_bsr.c.

10.33.2.4 fasp_ilu_dbsr_setup_omp()

```
SHORT fasp_ilu_dbsr_setup_omp (
            dBSRmat * A,
            ILU_data * iludata,
            ILU_param * iluparam )
```

Multi-threads parallel ILU decoposition of a BSR matrix A based on graph coloring.

Parameters

A	Pointer to dBSRmat matrix
iludata	Pointer to ILU_data
iluparam	Pointer to ILU_param

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Zheng Li

Date

12/04/2016

Note

Only works for 1, 2, 3 nb (Zheng)

Definition at line 973 of file ilu_setup_bsr.c.

10.34 ilu_setup_csr.c File Reference

Setup incomplete LU decomposition for dCSRmat matrices.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void **iluk_** (const INT ∗n, REAL ∗a, INT ∗ja, INT ∗ia, INT ∗lfil, REAL ∗alu, INT ∗jlu, INT ∗iwk, INT ∗ierr, INT ∗nzlu)
- void **ilut_** (const INT ∗n, REAL ∗a, INT ∗ja, INT ∗ia, INT ∗lfil, const REAL ∗droptol, REAL ∗alu, INT ∗jlu, INT ∗iwk, INT ∗ierr, INT ∗nz)
- void **ilutp_** (const INT ∗n, REAL ∗a, INT ∗ja, INT ∗ia, INT ∗lfil, const REAL ∗droptol, const REAL ∗permtol, const INT ∗mbloc, REAL ∗alu, INT ∗jlu, INT ∗iwk, INT ∗ierr, INT ∗nz)
- SHORT fasp_ilu_dcsr_setup (dCSRmat ∗A, ILU_data ∗iludata, ILU_param ∗iluparam)

  Get ILU decomposition of a CSR matrix A.

10.34.1 Detailed Description

Setup incomplete LU decomposition for dCSRmat matrices.

10.34.2 Function Documentation

10.34.2.1 fasp_ilu_dcsr_setup()

```
SHORT fasp_ilu_dcsr_setup (
            dCSRmat * A,
            ILU_data * iludata,
            ILU_param * iluparam )
```

Get ILU decomposition of a CSR matrix A.

Parameters

A	Pointer to dCSRmat matrix
iludata	Pointer to ILU_data
iluparam	Pointer to ILU_param

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Shiquan Zhang Xiaozhe Hu

Date

12/27/2009

Definition at line 50 of file ilu_setup_csr.c.

10.35 ilu_setup_str.c File Reference

Setup incomplete LU decomposition for dSTRmat matrices.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_ilu_dstr_setup0 (dSTRmat ∗A, dSTRmat ∗LU)

    Get ILU(0) decomposition of a structured matrix A.
- void fasp_ilu_dstr_setup1 (dSTRmat ∗A, dSTRmat ∗LU)

    Get ILU(1) decoposition of a structured matrix A.

10.35.1 Detailed Description

Setup incomplete LU decomposition for dSTRmat matrices.

10.35.2 Function Documentation

10.35.2.1 fasp_ilu_dstr_setup0()

```
void fasp_ilu_dstr_setup0 (
            dSTRmat * A,
            dSTRmat * LU )
```

Get ILU(0) decomposition of a structured matrix A.

Parameters

A	Pointer to dSTRmat
LU	Pointer to ILU structured matrix of REAL type

Author

Shiquan Zhang, Xiaozhe Hu

Date

11/08/2010

Note

Only works for 5 bands 2D and 7 bands 3D matrix with default offsets (order can be arbitrary)!

Definition at line 28 of file ilu_setup_str.c.

10.35.2.2 fasp_ilu_dstr_setup1()

```
void fasp_ilu_dstr_setup1 (
            dSTRmat * A,
            dSTRmat * LU )
```

Get ILU(1) decoposition of a structured matrix A.

Parameters

A	Pointer to oringinal structured matrix of REAL type
LU	Pointer to ILU structured matrix of REAL type

Author

Shiquan Zhang, Xiaozhe Hu

Date

11/08/2010

Note

put L and U in a STR matrix and it has the following structure: the diag is d, the offdiag of L are alpha1 to alpha6, the offdiag of U are beta1 to beta6

Only works for 5 bands 2D and 7 bands 3D matrix with default offsets

Definition at line 322 of file ilu_setup_str.c.

10.36 init.c File Reference

Initialize important data structures.

```
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_precond_data_null (precond_data ∗pcdata)
  Initialize precond_data.
- AMG_data ∗ fasp_amg_data_create (SHORT max_levels)
  Create and initialize AMG_data for classical and SA AMG.
- AMG_data_bsr ∗ fasp_amg_data_bsr_create (SHORT max_levels)
  Create and initialize AMG_data data sturcture for AMG/SAMG (BSR format)
- void fasp_ilu_data_alloc (const INT iwk, const INT nwork, ILU_data ∗iludata)
  Allocate workspace for ILU factorization.
- void fasp_Schwarz_data_free (Schwarz_data ∗Schwarz)
  Free Schwarz_data data memeory space.
- void fasp_amg_data_free (AMG_data ∗mgl, AMG_param ∗param)
  Free AMG_data data memeory space.
- void fasp_amg_data_bsr_free (AMG_data_bsr ∗mgl)
  Free AMG_data_bsr data memeory space.
- void fasp_ilu_data_free (ILU_data ∗ILUdata)
  Create ILU_data sturcture.
- void fasp_ilu_data_null (ILU_data ∗ILUdata)
  Initialize ILU data.
- void fasp_precond_null (precond ∗pcdata)
  Initialize precond data.

10.36.1 Detailed Description

Initialize important data structures.

Note

Every structures should be initialized before usage.

10.36.2 Function Documentation

10.36.2.1 fasp_amg_data_bsr_create()

```
AMG_data_bsr * fasp_amg_data_bsr_create (
            SHORT max_levels )
```

Create and initialize AMG_data data sturcture for AMG/SAMG (BSR format)

Parameters

max_levels	Max number of levels allowed

Returns

Pointer to the AMG_data data structure

Author

Xiaozhe Hu

Date

08/07/2011

Definition at line 86 of file init.c.

10.36.2.2 fasp_amg_data_bsr_free()

```
void fasp_amg_data_bsr_free (
            AMG_data_bsr * mgl )
```

Free AMG_data_bsr data memeory space.

Parameters

mgl	Pointer to the AMG_data_bsr

Author

Xiaozhe Hu

Date

2013/02/13

Definition at line 257 of file init.c.

10.36.2.3 fasp_amg_data_create()

```
AMG_data * fasp_amg_data_create (
            SHORT max_levels )
```

Create and initialize AMG_data for classical and SA AMG.

Parameters

max_levels	Max number of levels allowed

Returns

Pointer to the AMG_data data structure

Author

Chensong Zhang

Date

2010/04/06

Definition at line 56 of file init.c.

10.36.2.4 fasp_amg_data_free()

```
void fasp_amg_data_free (
            AMG_data * mgl,
            AMG_param * param )
```

Free AMG_data data memeory space.

Parameters

mgl	Pointer to the AMG_data
param	Pointer to AMG parameters

Author

Chensong Zhang

Date

2010/04/06

Modified by Chensong Zhang on 05/05/2013: Clean up param as well! Modified by Hongxuan Zhang on 12/15/2015: free internal memory for Intel MKL PARDISO.

Definition at line 185 of file init.c.

10.36.2.5 fasp_ilu_data_alloc()

```
void fasp_ilu_data_alloc (
            const INT iwk,
            const INT nwork,
            ILU_data * iludata )
```

Allocate workspace for ILU factorization.

Parameters

iwk	Size of the index array
nwork	Size of the work array
iludata	Pointer to the ILU_data

Author

Chensong Zhang

Date

2010/04/06

Definition at line 118 of file init.c.

10.36.2.6 fasp_ilu_data_free()

```
void fasp_ilu_data_free (
            ILU_data * ILUdata )
```

Create ILU_data sturcture.

Parameters

ILUdata	Pointer to ILU_data

Author

Chensong Zhang

Date

2010/04/03

Definition at line 301 of file init.c.

10.36.2.7 fasp_ilu_data_null()

```
void fasp_ilu_data_null (
            ILU_data * ILUdata )
```

Initialize ILU data.

Parameters

ILUdata	Pointer to ILU_data

Author

Chensong Zhang

Date

2010/03/23

Definition at line 326 of file init.c.

10.36.2.8 fasp_precond_data_null()

```
void fasp_precond_data_null (
            precond_data * pcdata )
```

Initialize precond_data.

Parameters

pcdata	Preconditioning data structure

Author

Chensong Zhang

Date

2010/03/23

Definition at line 25 of file init.c.

10.36.2.9 fasp_precond_null()

```
void fasp_precond_null (
            precond * pcdata )
```

Initialize precond data.

Parameters

pcdata	Pointer to precond

Author

Chensong Zhang

Date

2010/03/23

Definition at line 342 of file init.c.

10.36.2.10 fasp_Schwarz_data_free()

```
void fasp_Schwarz_data_free (
            Schwarz_data * Schwarz )
```

Free Schwarz_data data memeory space.

Parameters

∗*Schwarz*	pointer to the AMG_data data

Author

Xiaozhe Hu

Date

2010/04/06

Definition at line 147 of file init.c.

10.37 input.c File Reference

Read input parameters.

```
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- SHORT fasp_param_check (input_param *inparam)

  Simple check on input parameters.
- void fasp_param_input (const char *fname, input_param *inparam)

  Read input parameters from disk file.

10.37.1 Detailed Description

Read input parameters.

10.37.2 Function Documentation

10.37.2.1 fasp_param_check()

```
SHORT fasp_param_check (
            input_param * inparam )
```

Simple check on input parameters.

Parameters

inparam	Input parameters

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Chensong Zhang

Date

09/29/2013

Definition at line 25 of file input.c.

10.37.2.2 fasp_param_input()

```
void fasp_param_input (
            const char * fname,
            input_param * inparam )
```

Read input parameters from disk file.

Parameters

fname	File name for input file
inparam	Input parameters

Author

Chensong Zhang

Date

03/20/2010

Modified by Xiaozhe Hu on 01/23/2011: add AMLI cycle Modified by Chensong Zhang on 01/10/2012 Modified by Ludmil Zikatanov on 02/15/2013 Modified by Chensong Zhang on 05/10/2013: add a new input. Modified by Chensong Zhang on 03/23/2015: skip unknown keyword.

Definition at line 102 of file input.c.

10.38 interface_mumps.c File Reference

Interface to MUMPS direct solvers.

```
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Macros

- #define ICNTL(I) icntl[(I)-1]

Functions

- int fasp_solver_mumps (dCSRmat ∗ptrA, dvector ∗b, dvector ∗u, const SHORT prtlvl)

  Solve Ax=b by MUMPS directly.
- int fasp_solver_mumps_steps (dCSRmat ∗ptrA, dvector ∗b, dvector ∗u, Mumps_data ∗mumps)

  Solve Ax=b by MUMPS in three steps.

10.38.1 Detailed Description

Interface to MUMPS direct solvers.

Reference for MUMPS: http://mumps.enseeiht.fr/

10.38.2 Macro Definition Documentation

10.38.2.1 ICNTL

```
#define ICNTL(
              I ) icntl[(I)-1]
```

macro s.t. indices match documentation

Definition at line 17 of file interface_mumps.c.

10.38.3 Function Documentation

10.38.3.1 fasp_solver_mumps()

```
int fasp_solver_mumps (
             dCSRmat * ptrA,
             dvector * b,
             dvector * u,
             const SHORT prtlvl )
```

Solve Ax=b by MUMPS directly.

Parameters

ptrA	Pointer to a dCSRmat matrix
b	Pointer to the dvector of right-hand side term
u	Pointer to the dvector of solution
prtlvl	Output level

Author

Chunsheng Feng

Date

02/27/2013

Modified by Chensong Zhang on 02/27/2013 for new FASP function names.

Definition at line 39 of file interface_mumps.c.

10.38.3.2 fasp_solver_mumps_steps()

```
int fasp_solver_mumps_steps (
            dCSRmat * ptrA,
            dvector * b,
            dvector * u,
            Mumps_data * mumps )
```

Solve Ax=b by MUMPS in three steps.

Parameters

ptrA	Pointer to a dCSRmat matrix
b	Pointer to the dvector of right-hand side term
u	Pointer to the dvector of solution
mumps	Pointer to MUMPS data

Author

Chunsheng Feng

Date

02/27/2013

Modified by Chensong Zhang on 02/27/2013 for new FASP function names. Modified by Zheng Li on 10/10/2014 to adjust input parameters.

Definition at line 169 of file interface_mumps.c.

10.39 interface_pardiso.c File Reference

Interface to Intel MKL PARDISO direct solvers.

```
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- INT fasp_solver_pardiso (dCSRmat ∗ptrA, dvector ∗b, dvector ∗u, const SHORT prtlvl)
  Solve Ax=b by PARDISO directly. Each row of A should be in ascending order w.r.t. column indices.

10.39.1 Detailed Description

Interface to Intel MKL PARDISO direct solvers.

Reference for Intel MKL PARDISO: https://software.intel.com/en-us/node/470282

10.39.2 Function Documentation

10.39.2.1 fasp_solver_pardiso()

```
int fasp_solver_pardiso (
            dCSRmat * ptrA,
            dvector * b,
            dvector * u,
            const SHORT prtlvl )
```

Solve Ax=b by PARDISO directly. Each row of A should be in ascending order w.r.t. column indices.

Parameters

ptrA	Pointer to a dCSRmat matrix
b	Pointer to the dvector of right-hand side term
u	Pointer to the dvector of solution
prtlvl	Output level

Author

Hongxuan Zhang

Date

11/28/2015

Definition at line 38 of file interface_pardiso.c.

10.40 interface_samg.c File Reference

Interface to SAMG solvers.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void dvector2SAMGInput (dvector ∗vec, char ∗filename)

  Write a dvector to disk file in SAMG format (coordinate format)
- INT dCSRmat2SAMGInput (dCSRmat ∗A, char ∗filefrm, char ∗fileamg)

  Write SAMG Input data from a sparse matrix of CSR format.

10.40.1 Detailed Description

Interface to SAMG solvers.

Reference for SAMG: http://www.scai.fraunhofer.de/geschaeftsfelder/nuso/produkte/samg.↩
html

Warning

This interface has *only* been tested for SAMG24a1 (2010 version)!

10.40.2 Function Documentation

10.40.2.1 dCSRmat2SAMGInput()

```
INT dCSRmat2SAMGInput (
            dCSRmat * A,
            char * filefrm,
            char * fileamg )
```

Write SAMG Input data from a sparse matrix of CSR format.

Parameters

A	Pointer to the dCSRmat matrix
filefrm	Name of the .frm file
fileamg	Name of the .amg file

Author

Zhiyang Zhou

Date

2010/08/25

Definition at line 59 of file interface_samg.c.

10.40.2.2 dvector2SAMGInput()

```
void dvector2SAMGInput (
            dvector * vec,
            char * filename )
```

Write a dvector to disk file in SAMG format (coordinate format)

Parameters

vec	Pointer to the dvector
filename	File name for input

Author

Zhiyang Zhou

Date

08/25/2010

Definition at line 30 of file interface_samg.c.

10.41 interface_superlu.c File Reference

Interface to SuperLU direct solvers.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- int fasp_solver_superlu (dCSRmat ∗ptrA, dvector ∗b, dvector ∗u, const SHORT prtlvl)
  Solve Au=b by SuperLU.

10.41.1 Detailed Description

Interface to SuperLU direct solvers.

Reference for SuperLU: http://crd-legacy.lbl.gov/~xiaoye/SuperLU/

10.41.2 Function Documentation

10.41.2.1 fasp_solver_superlu()

```
int fasp_solver_superlu (
            dCSRmat * ptrA,
            dvector * b,
            dvector * u,
            const SHORT prtlvl )
```

Solve Au=b by SuperLU.

Parameters

ptrA	Pointer to a dCSRmat matrix
b	Pointer to the dvector of right-hand side term
u	Pointer to the dvector of solution
prtlvl	Output level

Author

Xiaozhe Hu

Date

11/05/09

Modified by Chensong Zhang on 11/01/2012 for new FASP function names. Modified by Chensong Zhang on 02/27/2013 for new FASP function names.

Definition at line 40 of file interface_superlu.c.

10.42 interface_umfpack.c File Reference

Interface to UMFPACK direct solvers.

```
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- INT fasp_solver_umfpack (dCSRmat *ptrA, dvector *b, dvector *u, const SHORT prtlvl)
  Solve Au=b by UMFpack.

10.42.1 Detailed Description

Interface to UMFPACK direct solvers.

Reference for SuiteSparse: http://faculty.cse.tamu.edu/davis/suitesparse.html

10.42.2 Function Documentation

10.42.2.1 fasp_solver_umfpack()

```
INT fasp_solver_umfpack (
            dCSRmat * ptrA,
            dvector * b,
            dvector * u,
            const SHORT prtlvl )
```

Solve Au=b by UMFpack.

Parameters

ptrA	Pointer to a dCSRmat matrix
b	Pointer to the dvector of right-hand side term
u	Pointer to the dvector of solution
prtlvl	Output level

Author

Chensong Zhang

Date

05/20/2010

Modified by Chensong Zhang on 02/27/2013 for new FASP function names.

Definition at line 37 of file interface_umfpack.c.

10.43 interpolation.c File Reference

Interpolation operators for AMG.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_amg_interp (dCSRmat ∗A, ivector ∗vertices, dCSRmat ∗P, iCSRmat ∗S, AMG_param ∗param)

    Generate interpolation operator P.
- void fasp_amg_interp1 (dCSRmat ∗A, ivector ∗vertices, dCSRmat ∗P, AMG_param ∗param, iCSRmat ∗S, INT ∗icor_ysk)

    Generate interpolation operator P.
- void fasp_amg_interp_trunc (dCSRmat ∗P, AMG_param ∗param)

    Truncation step for prolongation operators.

10.43.1 Detailed Description

Interpolation operators for AMG.

Note

Ref U. Trottenberg, C. W. Oosterlee, and A. Schuller "Multigrid (Appendix A: An Intro to Algebraic Multigrid)" Academic Press Inc., San Diego, CA, 2001 With contributions by A. Brandt, P. Oswald and K. Stuben.

10.43.2 Function Documentation

10.43.2.1 fasp_amg_interp()

```
void fasp_amg_interp (
            dCSRmat * A,
            ivector * vertices,
            dCSRmat * P,
            iCSRmat * S,
            AMG_param * param )
```

Generate interpolation operator P.

Parameters

A	Pointer to dCSRmat: the coefficient matrix (index starts from 0)
vertices	Indicator vector for the C/F splitting of the variables
P	Prolongation (input: nonzero pattern, output: prolongation)
S	Strong connection matrix
param	AMG parameters

Author

Xuehai Huang, Chensong Zhang

Date

04/04/2010

Modified by Xiaozhe Hu on 05/23/2012: add S as input Modified by Chensong Zhang on 09/12/2012: clean up and debug interp_RS Modified by Chensong Zhang on 05/14/2013: reconstruct the code

Definition at line 48 of file interpolation.c.

10.43.2.2 fasp_amg_interp1()

```
void fasp_amg_interp1 (
            dCSRmat * A,
            ivector * vertices,
            dCSRmat * P,
            AMG_param * param,
            iCSRmat * S,
            INT * icor_ysk )
```

Generate interpolation operator P.

Parameters

A	Pointer to dCSRmat: the coefficient matrix (index starts from 0)
vertices	Indicator vector for the C/F splitting of the variables
P	Prolongation (input: nonzero pattern, output: prolongation)
S	Strong connection matrix
param	AMG parameters
icor_ysk	Indices of coarse nodes in fine grid

Returns

FASP_SUCCESS or error message

Author

Chunsheng Feng, Xiaoqiang Yue

Date

03/01/2011

Modified by Chensong Zhang on 05/14/2013: reconstruct the code

Definition at line 105 of file interpolation.c.

10.43.2.3 fasp_amg_interp_trunc()

```
void fasp_amg_interp_trunc (
            dCSRmat * P,
            AMG_param * param )
```

Truncation step for prolongation operators.

Parameters

P	Prolongation (input: full, output: truncated)
param	Pointer to AMG_param: AMG parameters

Author

Chensong Zhang

Date

05/14/2013

Originally by Xuehai Huang, Chensong Zhang on 01/31/2009 Modified by Chunsheng Feng, Xiaoqiang Yue on 05/23/2012: add OMP support Modified by Chensong Zhang on 05/14/2013: rewritten

Definition at line 159 of file interpolation.c.

10.44 interpolation_em.c File Reference

Interpolation operators for AMG based on energy-min.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_amg_interp_em (dCSRmat ∗A, ivector ∗vertices, dCSRmat ∗P, AMG_param ∗param)

  Energy-min interpolation.

10.44.1 Detailed Description

Interpolation operators for AMG based on energy-min.

Note

Ref J. Xu and L. Zikatanov "On An Energy Minimizing Basis in Algebraic Multigrid Methods" Computing and visualization in sciences, 2003

10.44.2 Function Documentation

10.44.2.1 fasp_amg_interp_em()

```
void fasp_amg_interp_em (
            dCSRmat * A,
            ivector * vertices,
            dCSRmat * P,
            AMG_param * param )
```

Energy-min interpolation.

Parameters

A	Pointer to dCSRmat: the coefficient matrix (index starts from 0)
vertices	Pointer to the indicator of CF splitting on fine or coarse grid
P	Pointer to the dCSRmat matrix of resulted interpolation
param	Pointer to AMG_param: AMG parameters

Author

Shuo Zhang, Xuehai Huang

Date

04/04/2010

Modified by Chunsheng Feng, Zheng Li on 10/17/2012: add OMP support Modified by Chensong Zhang on 05/14/2013: reconstruct the code

Definition at line 49 of file interpolation_em.c.

10.45 io.c File Reference

Matrix/vector input/output subroutines.

```
#include "fasp.h"
#include "fasp_functs.h"
#include "hb_io.h"
```

Functions

- void fasp_dcsrvec1_read (const char ∗filename, dCSRmat ∗A, dvector ∗b)

  Read A and b from a SINGLE disk file.
- void fasp_dcsrvec2_read (const char ∗filemat, const char ∗filerhs, dCSRmat ∗A, dvector ∗b)

  Read A and b from two disk files.
- void fasp_dcsr_read (const char ∗filename, dCSRmat ∗A)

  Read A from matrix disk file in IJ format.
- void fasp_dcoo_read (const char ∗filename, dCSRmat ∗A)

  Read A from matrix disk file in IJ format – indices starting from 0.
- void fasp_dcoo1_read (const char ∗filename, dCOOmat ∗A)

  Read A from matrix disk file in IJ format – indices starting from 1.
- void fasp_dcoo_shift_read (const char ∗filename, dCSRmat ∗A)

  Read A from matrix disk file in IJ format – indices starting from 0.
- void fasp_dmtx_read (const char ∗filename, dCSRmat ∗A)

  Read A from matrix disk file in MatrixMarket general format.

- void fasp_dmtxsym_read (const char ∗filename, dCSRmat ∗A)

    Read A from matrix disk file in MatrixMarket sym format.
- void fasp_dstr_read (const char ∗filename, dSTRmat ∗A)

    Read A from a disk file in dSTRmat format.
- void fasp_dbsr_read (const char ∗filename, dBSRmat ∗A)

    Read A from a disk file in dBSRmat format.
- void fasp_dvecind_read (const char ∗filename, dvector ∗b)

    Read b from matrix disk file.
- void fasp_dvec_read (const char ∗filename, dvector ∗b)

    Read b from a disk file in array format.
- void fasp_ivecind_read (const char ∗filename, ivector ∗b)

    Read b from matrix disk file.
- void fasp_ivec_read (const char ∗filename, ivector ∗b)

    Read b from a disk file in array format.
- void fasp_dcsrvec1_write (const char ∗filename, dCSRmat ∗A, dvector ∗b)

    Write A and b to a SINGLE disk file.
- void fasp_dcsrvec2_write (const char ∗filemat, const char ∗filerhs, dCSRmat ∗A, dvector ∗b)

    Write A and b to two disk files.
- void fasp_dcoo_write (const char ∗filename, dCSRmat ∗A)

    Write a matrix to disk file in IJ format (coordinate format)
- void fasp_dstr_write (const char ∗filename, dSTRmat ∗A)

    Write a dSTRmat to a disk file.
- void fasp_dbsr_write (const char ∗filename, dBSRmat ∗A)

    Write a dBSRmat to a disk file.
- void fasp_dvec_write (const char ∗filename, dvector ∗vec)

    Write a dvector to disk file.
- void fasp_dvecind_write (const char ∗filename, dvector ∗vec)

    Write a dvector to disk file in coordinate format.
- void fasp_ivec_write (const char ∗filename, ivector ∗vec)

    Write a ivector to disk file in coordinate format.
- void fasp_dvec_print (INT n, dvector ∗u)

    Print first n entries of a vector of REAL type.
- void fasp_ivec_print (INT n, ivector ∗u)

    Print first n entries of a vector of INT type.
- void fasp_dcsr_print (dCSRmat ∗A)

    Print out a dCSRmat matrix in coordinate format.
- void fasp_dcoo_print (dCOOmat ∗A)

    Print out a dCOOmat matrix in coordinate format.
- void fasp_dbsr_print (dBSRmat ∗A)

    Print out a dBSRmat matrix in coordinate format.
- void fasp_dbsr_write_coo (const char ∗filename, const dBSRmat ∗A)

    Print out a dBSRmat matrix in coordinate format for matlab spy.
- void fasp_dcsr_write_coo (const char ∗filename, const dCSRmat ∗A)

    Print out a dCSRmat matrix in coordinate format for matlab spy.
- void fasp_dstr_print (dSTRmat ∗A)

    Print out a dSTRmat matrix in coordinate format.
- void fasp_matrix_read (const char ∗filename, void ∗A)

Read matrix from different kinds of formats from both ASCII and binary files.
- void fasp_matrix_read_bin (const char ∗filename, void ∗A)

  Read matrix in binary format.
- void fasp_matrix_write (const char ∗filename, void ∗A, INT flag)

  write matrix from different kinds of formats from both ASCII and binary files
- void fasp_vector_read (const char ∗filerhs, void ∗b)

  Read RHS vector from different kinds of formats from both ASCII and binary files.
- void fasp_vector_write (const char ∗filerhs, void ∗b, INT flag)

  write RHS vector from different kinds of formats in both ASCII and binary files
- void fasp_hb_read (const char ∗input_file, dCSRmat ∗A, dvector ∗b)

  Read matrix and right-hans side from a HB format file.

Variables

- INT ilength
- INT dlength

10.45.1 Detailed Description

Matrix/vector input/output subroutines.

Note

Read, write or print a matrix or a vector in various formats.

10.45.2 Function Documentation

10.45.2.1 fasp_dbsr_print()

```
void fasp_dbsr_print (
            dBSRmat * A )
```

Print out a dBSRmat matrix in coordinate format.

Parameters

A	Pointer to the dBSRmat matrix A

Author

Ziteng Wang

Date

12/24/2012

Modified by Chunsheng Feng on 11/16/2013

Definition at line 1439 of file io.c.

10.45.2.2 fasp_dbsr_read()

```
void fasp_dbsr_read (
            const char * filename,
            dBSRmat * A )
```

Read A from a disk file in dBSRmat format.

Parameters

filename	File name for matrix A
A	Pointer to the dBSRmat A

Note

This routine reads a dBSRmat matrix from a disk file in the following format:
File format:

- ROW, COL, NNZ
- nb: size of each block
- storage_manner: storage manner of each block
- ROW+1: length of IA
- IA(i), i=0:ROW
- NNZ: length of JA
- JA(i), i=0:NNZ-1
- NNZ*nb*nb: length of val
- val(i), i=0:NNZ*nb*nb-1

Author

Xiaozhe Hu

Date

10/29/2010

Definition at line 690 of file io.c.

10.45.2.3 fasp_dbsr_write()

```
void fasp_dbsr_write (
            const char * filename,
            dBSRmat * A )
```

Write a dBSRmat to a disk file.

Parameters

filename	File name for A
A	Pointer to the dBSRmat matrix A

Note

```
The routine writes the specified REAL vector in BSR format.
Refer to the reading subroutine \ref fasp_dbsr_read.
```

Author

Shiquan Zhang

Date

10/29/2010

Definition at line 1197 of file io.c.

10.45.2.4 fasp_dbsr_write_coo()

```
void fasp_dbsr_write_coo (
            const char * filename,
            const dBSRmat * A )
```

Print out a dBSRmat matrix in coordinate format for matlab spy.

Parameters

filename	Name of file to write to
A	Pointer to the dBSRmat matrix A

Author

Chunsheng Feng

Date

11/14/2013

Modified by Chensong Zhang on 06/14/2014: Fix index problem.

Definition at line 1475 of file io.c.

10.45.2.5 fasp_dcoo1_read()

```
void fasp_dcoo1_read (
            const char * filename,
            dCOOmat * A )
```

Read A from matrix disk file in IJ format – indices starting from 1.

Parameters

filename	File name for matrix
A	Pointer to the COO matrix

Note

File format:

- nrow ncol nnz % number of rows, number of columns, and nnz
- i j a_ij % i, j a_ij in each line

difference between fasp_dcoo_read and this funciton is this function do not change to CSR format

Author

Xiaozhe Hu

Date

03/24/2013

Definition at line 369 of file io.c.

10.45.2.6 fasp_dcoo_print()

```
void fasp_dcoo_print (
            dCOOmat * A )
```

Print out a dCOOmat matrix in coordinate format.

Parameters

A	Pointer to the dCOOmat matrix A

Author

Ziteng Wang

Date

12/24/2012

Definition at line 1418 of file io.c.

10.45.2.7 fasp_dcoo_read()

```
void fasp_dcoo_read (
            const char * filename,
            dCSRmat * A )
```

Read A from matrix disk file in IJ format – indices starting from 0.

Parameters

filename	File name for matrix
A	Pointer to the CSR matrix

Note

File format:

- nrow ncol nnz % number of rows, number of columns, and nnz
- i j a_ij % i, j a_ij in each line

After reading, it converts the matrix to dCSRmat format.

Author

Xuehai Huang, Chensong Zhang

Date

03/29/2009

Definition at line 318 of file io.c.

10.45.2.8 fasp_dcoo_shift_read()

```
void fasp_dcoo_shift_read (
            const char * filename,
            dCSRmat * A )
```

Read A from matrix disk file in IJ format – indices starting from 0.

Parameters

filename	File name for matrix
A	Pointer to the CSR matrix

Note

File format:

- nrow ncol nnz % number of rows, number of columns, and nnz
- i j a_ij % i, j a_ij in each line

i and j suppose to start with index 1!!!
After read in, it shifts the index to C fashin and converts the matrix to dCSRmat format.

Author

Xiaozhe Hu

Date

04/01/2014

Definition at line 419 of file io.c.

10.45.2.9 fasp_dcoo_write()

```
void fasp_dcoo_write (
            const char * filename,
            dCSRmat * A )
```

Write a matrix to disk file in IJ format (coordinate format)

Parameters

A	pointer to the dCSRmat matrix
filename	char for vector file name

Note

```
The routine writes the specified REAL vector in COO format.
Refer to the reading subroutine \ref fasp_dcoo_read.
```

File format:

- The first line of the file gives the number of rows, the number of columns, and the number of nonzeros.
- Then gives nonzero values in i j a(i,j) format.

Author

Chensong Zhang

Date

03/29/2009

Definition at line 1098 of file io.c.

10.45.2.10 fasp_dcsr_print()

```
void fasp_dcsr_print (
            dCSRmat * A )
```

Print out a dCSRmat matrix in coordinate format.

Parameters

A	Pointer to the dCSRmat matrix A

Author

Xuehai Huang

Date

03/29/2009

Definition at line 1396 of file io.c.

10.45.2.11 fasp_dcsr_read()

```
void fasp_dcsr_read (
            const char * filename,
            dCSRmat * A )
```

Read A from matrix disk file in IJ format.

Parameters

∗*filename*	char for matrix file name
∗*A*	pointer to the CSR matrix

Author

Ziteng Wang

Date

12/25/2012

Definition at line 257 of file io.c.

10.45.2.12 fasp_dcsr_write_coo()

```
void fasp_dcsr_write_coo (
            const char * filename,
            const dCSRmat * A )
```

Print out a dCSRmat matrix in coordinate format for matlab spy.

Parameters

filename	Name of file to write to
A	Pointer to the dCSRmat matrix A

Author

Chunsheng Feng

Date

11/14/2013

Definition at line 1525 of file io.c.

10.45.2.13 fasp_dcsrvec1_read()

```
void fasp_dcsrvec1_read (
            const char * filename,
            dCSRmat * A,
            dvector * b )
```

Read A and b from a SINGLE disk file.

Parameters

filename	File name
A	Pointer to the CSR matrix
b	Pointer to the dvector

Note

This routine reads a dCSRmat matrix and a dvector vector from a single disk file.

```
The difference between this and fasp_dcoovec_read is that this
routine support non-square matrices.
```

File format:

- nrow ncol % number of rows and number of columns
- ia(j), j=0:nrow % row index
- ja(j), j=0:nnz-1 % column index
- a(j), j=0:nnz-1 % entry value
- n % number of entries
- b(j), j=0:n-1 % entry value

Author

Xuehai Huang

Date

03/29/2009

Modified by Chensong Zhang on 03/14/2012

Definition at line 86 of file io.c.

10.45.2.14 fasp_dcsrvec1_write()

```
void fasp_dcsrvec1_write (
            const char * filename,
            dCSRmat * A,
            dvector * b )
```

Write A and b to a SINGLE disk file.

Parameters

filename	File name
A	Pointer to the CSR matrix
b	Pointer to the dvector

Note

This routine writes a dCSRmat matrix and a dvector vector to a single disk file.
File format:

- nrow ncol % number of rows and number of columns

- ia(j), j=0:nrow % row index
- ja(j), j=0:nnz-1 % column index
- a(j), j=0:nnz-1 % entry value
- n % number of entries
- b(j), j=0:n-1 % entry value

Author

Feiteng Huang

Date

05/19/2012

Modified by Chensong on 12/26/2012

Definition at line 949 of file io.c.

10.45.2.15 fasp_dcsrvec2_read()

```
void fasp_dcsrvec2_read (
            const char * filemat,
            const char * filerhs,
            dCSRmat * A,
            dvector * b )
```

Read A and b from two disk files.

Parameters

filemat	File name for matrix
filerhs	File name for right-hand side
A	Pointer to the dCSR matrix
b	Pointer to the dvector

Note

`This routine reads a dCSRmat matrix and a dvector vector from a disk file.`

CSR matrix file format:

- nrow % number of columns (rows)
- ia(j), j=0:nrow % row index
- ja(j), j=0:nnz-1 % column index
- a(j), j=0:nnz-1 % entry value

RHS file format:

- n % number of entries

- b(j), j=0:nrow-1 % entry value

Indices start from 1, NOT 0!!!

Author

Zhiyang Zhou

Date

2010/08/06

Modified by Chensong Zhang on 2011/03/01 Modified by Chensong Zhang on 2012/01/05

Definition at line 178 of file io.c.

10.45.2.16 fasp_dcsrvec2_write()

```
void fasp_dcsrvec2_write (
            const char * filemat,
            const char * filerhs,
            dCSRmat * A,
            dvector * b )
```

Write A and b to two disk files.

Parameters

filemat	File name for matrix
filerhs	File name for right-hand side
A	Pointer to the dCSR matrix
b	Pointer to the dvector

Note

```
This routine writes a dCSRmat matrix and a dvector vector to two disk files.
```

CSR matrix file format:

- nrow % number of columns (rows)
- ia(j), j=0:nrow % row index
- ja(j), j=0:nnz-1 % column index
- a(j), j=0:nnz-1 % entry value

RHS file format:

- n % number of entries
- b(j), j=0:nrow-1 % entry value

Indices start from 1, NOT 0!!!

Author

Feiteng Huang

Date

05/19/2012

Definition at line 1027 of file io.c.

10.45.2.17 fasp_dmtx_read()

```
void fasp_dmtx_read (
            const char * filename,
            dCSRmat * A )
```

Read A from matrix disk file in MatrixMarket general format.

Parameters

filename	File name for matrix
A	Pointer to the CSR matrix

Note

File format: This routine reads a MatrixMarket general matrix from a mtx file. And it converts the matrix to dCS←Rmat format. For details of mtx format, please refer to `http://math.nist.gov/MatrixMarket/`.
Indices start from 1, NOT 0!!!

Author

Chensong Zhang

Date

09/05/2011

Definition at line 471 of file io.c.

10.45.2.18 fasp_dmtxsym_read()

```
void fasp_dmtxsym_read (
            const char * filename,
            dCSRmat * A )
```

Read A from matrix disk file in MatrixMarket sym format.

Parameters

filename	File name for matrix
A	Pointer to the CSR matrix

Note

File format: This routine reads a MatrixMarket symmetric matrix from a mtx file. And it converts the matrix to dCSRmat format. For details of mtx format, please refer to `http://math.nist.gov/MatrixMarket/`.

```
Indices start from 1, NOT 0!!!
```

Author

Chensong Zhang

Date

09/02/2011

Definition at line 533 of file io.c.

10.45.2.19 fasp_dstr_print()

```
void fasp_dstr_print (
            dSTRmat * A )
```

Print out a dSTRmat matrix in coordinate format.

Parameters

A	Pointer to the dSTRmat matrix A

Author

Ziteng Wang

Date

12/24/2012

Definition at line 1564 of file io.c.

10.45.2.20 fasp_dstr_read()

```
void fasp_dstr_read (
            const char * filename,
            dSTRmat * A )
```

Read A from a disk file in dSTRmat format.

Parameters

filename	File name for the matrix
A	Pointer to the dSTRmat

Note

This routine reads a dSTRmat matrix from a disk file. After done, it converts the matrix to dCSRmat format.
File format:

- nx, ny, nz
- nc: number of components
- nband: number of bands
- n: size of diagonal, you must have diagonal
- diag(j), j=0:n-1
- offset, length: offset and length of off-diag1
- offdiag(j), j=0:length-1

Author

Xuehai Huang

Date

03/29/2009

Definition at line 610 of file io.c.

10.45.2.21 fasp_dstr_write()

```
void fasp_dstr_write (
            const char * filename,
            dSTRmat * A )
```

Write a dSTRmat to a disk file.

Parameters

filename	File name for A
A	Pointer to the dSTRmat matrix A

Note

```
The routine writes the specified REAL vector in STR format.
Refer to the reading subroutine \ref fasp_dstr_read.
```

Author

Shiquan Zhang

Date

03/29/2010

Definition at line 1138 of file io.c.

10.45.2.22 fasp_dvec_print()

```
void fasp_dvec_print (
            INT n,
            dvector * u )
```

Print first n entries of a vector of REAL type.

Parameters

n	An interger (if n=0, then print all entries)
u	Pointer to a dvector

Author

Chensong Zhang

Date

03/29/2009

Definition at line 1357 of file io.c.

10.45.2.23 fasp_dvec_read()

```
void fasp_dvec_read (
            const char * filename,
            dvector * b )
```

Read b from a disk file in array format.

Parameters

filename	File name for vector b
b	Pointer to the dvector b (output)

Note

File Format:

- nrow
- val_j, j=0:nrow-1

Author

Chensong Zhang

Date

03/29/2009

Definition at line 809 of file io.c.

10.45.2.24 fasp_dvec_write()

```
void fasp_dvec_write (
            const char * filename,
            dvector * vec )
```

Write a dvector to disk file.

Parameters

vec	Pointer to the dvector
filename	File name

Author

Xuehai Huang

Date

03/29/2009

Definition at line 1252 of file io.c.

10.45.2.25 fasp_dvecind_read()

```
void fasp_dvecind_read (
            const char * filename,
            dvector * b )
```

Read b from matrix disk file.

Parameters

filename	File name for vector b
b	Pointer to the dvector b (output)

Note

File Format:

- nrow
- ind_j, val_j, j=0:nrow-1

Because the index is given, order is not important!

Author

Chensong Zhang

Date

03/29/2009

Definition at line 759 of file io.c.

10.45.2.26 fasp_dvecind_write()

```
void fasp_dvecind_write (
            const char * filename,
            dvector * vec )
```

Write a dvector to disk file in coordinate format.

Parameters

vec	Pointer to the dvector
filename	File name

Note

The routine writes the specified REAL vector in IJ format.

- The first line of the file is the length of the vector;
- After that, each line gives index and value of the entries.

Author

Xuehai Huang

Date

03/29/2009

Definition at line 1288 of file io.c.

10.45.2.27 fasp_hb_read()

```
fasp_hb_read (
            const char * input_file,
            dCSRmat * A,
            dvector * b )
```

Read matrix and right-hans side from a HB format file.

Parameters

input_file	File name of vector file
A	Pointer to the matrix
b	Pointer to the vector

Note

Modified from the c code hb_io_prb.c by John Burkardt

Author

Xiaoehe Hu

Date

05/30/2014

Definition at line 2059 of file io.c.

10.45.2.28 fasp_ivec_print()

```
void fasp_ivec_print (
            INT n,
            ivector * u )
```

Print first n entries of a vector of INT type.

Parameters

n	An interger (if n=0, then print all entries)
u	Pointer to an ivector

Author

Chensong Zhang

Date

03/29/2009

Definition at line 1377 of file io.c.

10.45.2.29 fasp_ivec_read()

```
void fasp_ivec_read (
            const char * filename,
            ivector * b )
```

Read b from a disk file in array format.

Parameters

filename	File name for vector b
b	Pointer to the dvector b (output)

Note

File Format:

- nrow
- val_j, j=0:nrow-1

Author

Xuehai Huang

Date

03/29/2009

Definition at line 899 of file io.c.

10.45.2.30 fasp_ivec_write()

```
void fasp_ivec_write (
            const char * filename,
            ivector * vec )
```

Write a ivector to disk file in coordinate format.

Parameters

vec	Pointer to the dvector
filename	File name

Note

The routine writes the specified INT vector in IJ format.

- The first line of the file is the length of the vector;
- After that, each line gives index and value of the entries.

Author

Xuehai Huang

Date

03/29/2009

Definition at line 1323 of file io.c.

10.45.2.31 fasp_ivecind_read()

```
void fasp_ivecind_read (
            const char * filename,
            ivector * b )
```

Read b from matrix disk file.

Parameters

filename	File name for vector b
b	Pointer to the dvector b (output)

Note

File Format:

- nrow
- ind_j, val_j ... j=0:nrow-1

Author

Chensong Zhang

Date

03/29/2009

Definition at line 859 of file io.c.

10.45.2.32 fasp_matrix_read()

```
fasp_matrix_read (
            const char * filemat,
            void * A )
```

Read matrix from different kinds of formats from both ASCII and binary files.

Parameters

filemat	File name of matrix file
A	Pointer to the matrix

Note

Flags for matrix file format:

- fileflag % fileflag = 1: binary, fileflag = 0000: ASCII
- formatflag % a 3-digit number for internal use, see below
- matrix % different types of matrix

Meaning of formatflag:

- matrixflag % first digit of formatflag
  - matrixflag = 1: CSR format
  - matrixflag = 2: BSR format
  - matrixflag = 3: STR format
  - matrixflag = 4: COO format
  - matrixflag = 5: MTX format
  - matrixflag = 6: MTX symmetrical format
- ilength % third digit of formatflag, length of INT
- dlength % fourth digit of formatflag, length of REAL

Author

Ziteng Wang

Date

12/24/2012

Modified by Chensong Zhang on 05/01/2013

Definition at line 1598 of file io.c.

10.45.2.33 fasp_matrix_read_bin()

```
void fasp_matrix_read_bin (
            const char * filemat,
            void * A )
```

Read matrix in binary format.

Parameters

filemat	File name of matrix file
A	Pointer to the matrix

Author

Xiaozhe Hu

Date

04/14/2013

Modified by Chensong Zhang on 05/01/2013: Use it to read binary files!!!

Definition at line 1704 of file io.c.

10.45.2.34 fasp_matrix_write()

```
fasp_matrix_write (
            const char * filemat,
            void * A,
            INT flag )
```

write matrix from different kinds of formats from both ASCII and binary files

Parameters

filemat	File name of matrix file
A	Pointer to the matrix
flag	Type of file and matrix, a 3-digit number

Note

Meaning of flag:

- fileflag % fileflag = 1: binary, fileflag = 0: ASCII
- matrixflag
  - matrixflag = 1: CSR format
  - matrixflag = 2: BSR format
  - matrixflag = 3: STR format

Matrix file format:

- fileflag % fileflag = 1: binary, fileflag = 0000: ASCII
- formatflag % a 3-digit number
- matrixflag % different kinds of matrix judged by formatflag

Author

Ziteng Wang

Date

12/24/2012

Definition at line 1778 of file io.c.

10.45.2.35 fasp_vector_read()

```
fasp_vector_read (
            const char * filerhs,
            void * b )
```

Read RHS vector from different kinds of formats from both ASCII and binary files.

Parameters

filerhs	File name of vector file
b	Pointer to the vector

Note

Matrix file format:

- fileflag % fileflag = 1: binary, fileflag = 0000: ASCII
- formatflag % a 3-digit number
- vector % different kinds of vector judged by formatflag

Meaning of formatflag:

- vectorflag % first digit of formatflag
  - vectorflag = 1: dvec format
  - vectorflag = 2: ivec format
  - vectorflag = 3: dvecind format
  - vectorflag = 4: ivecind format
- ilength % second digit of formatflag, length of INT
- dlength % third digit of formatflag, length of REAL

Author

Ziteng Wang

Date

12/24/2012

Definition at line 1872 of file io.c.

10.45.2.36 fasp_vector_write()

```
fasp_vector_write (
            const char * filerhs,
            void * b,
            INT flag )
```

write RHS vector from different kinds of formats in both ASCII and binary files

Parameters

filerhs	File name of vector file
b	Pointer to the vector
flag	Type of file and vector, a 2-digit number

Note

Meaning of the flags

- fileflag % fileflag = 1: binary, fileflag = 0: ASCII
- vectorflag

  - vectorflag = 1: dvec format
  - vectorflag = 2: ivec format
  - vectorflag = 3: dvecind format
  - vectorflag = 4: ivecind format

Matrix file format:

- fileflag % fileflag = 1: binary, fileflag = 0000: ASCII
- formatflag % a 2-digit number
- vectorflag % different kinds of vector judged by formatflag

Author

Ziteng Wang

Date

12/24/2012

Modified by Chensong Zhang on 05/02/2013: fix a bug when writing in binary format

Definition at line 1970 of file io.c.

10.45.3 Variable Documentation

10.45.3.1 dlength

```
INT dlength
```

Length of REAL in byte

Definition at line 14 of file io.c.

10.45.3.2 ilength

```
INT ilength
```

Length of INT in byte

Definition at line 13 of file io.c.

10.46 itsolver_blc.c File Reference

Iterative solvers for dBLCmat matrices.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_block.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_dblc_itsolver (dBLCmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, itsolver_param ∗itparam)

  Solve Ax = b by standard Krylov methods.
- INT fasp_solver_dblc_krylov (dBLCmat ∗A, dvector ∗b, dvector ∗x, itsolver_param ∗itparam)

  Solve Ax = b by standard Krylov methods.
- INT fasp_solver_dblc_krylov_block_3 (dBLCmat ∗A, dvector ∗b, dvector ∗x, itsolver_param ∗itparam, AMG_↩ param ∗amgparam, dCSRmat ∗A_diag)

  Solve Ax = b by standard Krylov methods.
- INT fasp_solver_dblc_krylov_block_4 (dBLCmat ∗A, dvector ∗b, dvector ∗x, itsolver_param ∗itparam, AMG_↩ param ∗amgparam, dCSRmat ∗A_diag)

  Solve Ax = b by standard Krylov methods.
- INT fasp_solver_dblc_krylov_sweeping (dBLCmat ∗A, dvector ∗b, dvector ∗x, itsolver_param ∗itparam, INT NumLayers, dBLCmat ∗Ai, dCSRmat ∗local_A, ivector ∗local_index)

  Solve Ax = b by standard Krylov methods.

10.46.1 Detailed Description

Iterative solvers for dBLCmat matrices.

10.46.2 Function Documentation

10.46.2.1 fasp_solver_dblc_itsolver()

```
INT fasp_solver_dblc_itsolver (
            dBLCmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            itsolver_param * itparam )
```

Solve Ax = b by standard Krylov methods.

Parameters

A	Pointer to the coeff matrix in dBLCmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
pc	Pointer to the preconditioning action
itparam	Pointer to parameters for iterative solvers

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

11/25/2010

Modified by Chunsheng Feng on 03/04/2016: add VBiCGstab solver

Definition at line 38 of file itsolver_blc.c.

10.46.2.2 fasp_solver_dblc_krylov()

```
INT fasp_solver_dblc_krylov (
            dBLCmat * A,
            dvector * b,
            dvector * x,
            itsolver_param * itparam )
```

Solve Ax = b by standard Krylov methods.

Parameters

A	Pointer to the coeff matrix in dBLCmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

07/18/2010

Definition at line 130 of file itsolver_blc.c.

10.46.2.3 fasp_solver_dblc_krylov_block_3()

```
INT fasp_solver_dblc_krylov_block_3 (
            dBLCmat * A,
            dvector * b,
            dvector * x,
            itsolver_param * itparam,
            AMG_param * amgparam,
            dCSRmat * A_diag )
```

Solve Ax = b by standard Krylov methods.

Parameters

A	Pointer to the coeff matrix in dBLCmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers
amgparam	Pointer to parameters for AMG solvers
A_diag	Digonal blocks of A

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

07/10/2014

Note

only works for 3by3 block dCSRmat problems!! – Xiaozhe Hu

Definition at line 184 of file itsolver_blc.c.

10.46.2.4 fasp_solver_dblc_krylov_block_4()

```
INT fasp_solver_dblc_krylov_block_4 (
            dBLCmat * A,
            dvector * b,
            dvector * x,
            itsolver_param * itparam,
            AMG_param * amgparam,
            dCSRmat * A_diag )
```

Solve Ax = b by standard Krylov methods.

Parameters

A	Pointer to the coeff matrix in dBLCmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers
amgparam	Pointer to parameters for AMG solvers
A_diag	Digonal blocks of A

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

07/06/2014

Note

only works for 4 by 4 block dCSRmat problems!! – Xiaozhe Hu

Definition at line 390 of file itsolver_blc.c.

10.46.2.5 fasp_solver_dblc_krylov_sweeping()

```
INT fasp_solver_dblc_krylov_sweeping (
            dBLCmat * A,
            dvector * b,
            dvector * x,
            itsolver_param * itparam,
            INT NumLayers,
            dBLCmat * Ai,
            dCSRmat * local_A,
            ivector * local_index )
```

Solve Ax = b by standard Krylov methods.

Parameters

A	Pointer to the coeff matrix in dBLCmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers
NumLayers	Number of layers used for sweeping preconditioner
Ai	Pointer to the coeff matrix for the preconditioner in dBLCmat format
local_A	Pointer to the local coeff matrices in the dCSRmat format
local_index	Pointer to the local index in ivector format

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

05/01/2014

Definition at line 516 of file itsolver_blc.c.

10.47 itsolver_bsr.c File Reference

Iterative solvers for dBSRmat matrices.

```
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_dbsr_itsolver (dBSRmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, itsolver_param ∗itparam)

  Solve Ax=b by preconditioned Krylov methods for BSR matrices.
- INT fasp_solver_dbsr_krylov (dBSRmat ∗A, dvector ∗b, dvector ∗x, itsolver_param ∗itparam)

  Solve Ax=b by standard Krylov methods for BSR matrices.
- INT fasp_solver_dbsr_krylov_diag (dBSRmat ∗A, dvector ∗b, dvector ∗x, itsolver_param ∗itparam)

  Solve Ax=b by diagonal preconditioned Krylov methods.
- INT fasp_solver_dbsr_krylov_ilu (dBSRmat ∗A, dvector ∗b, dvector ∗x, itsolver_param ∗itparam, ILU_param ∗iluparam)

Solve Ax=b by ILUs preconditioned Krylov methods.

- INT fasp_solver_dbsr_krylov_amg (dBSRmat ∗A, dvector ∗b, dvector ∗x, itsolver_param ∗itparam, AMG_param ∗amgparam)

  Solve Ax=b by AMG preconditioned Krylov methods.
- INT fasp_solver_dbsr_krylov_amg_nk (dBSRmat ∗A, dvector ∗b, dvector ∗x, itsolver_param ∗itparam, AMG_param ∗amgparam, dCSRmat ∗A_nk, dCSRmat ∗P_nk, dCSRmat ∗R_nk)

  Solve Ax=b by AMG with extra near kernel solve preconditioned Krylov methods.
- INT fasp_solver_dbsr_krylov_nk_amg (dBSRmat ∗A, dvector ∗b, dvector ∗x, itsolver_param ∗itparam, AMG_param ∗amgparam, const INT nk_dim, dvector ∗nk)

  Solve Ax=b by AMG preconditioned Krylov methods with extra kernal space.

10.47.1 Detailed Description

Iterative solvers for dBSRmat matrices.

10.47.2 Function Documentation

10.47.2.1 fasp_solver_dbsr_itsolver()

```
INT fasp_solver_dbsr_itsolver (
            dBSRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            itsolver_param * itparam )
```

Solve Ax=b by preconditioned Krylov methods for BSR matrices.

Parameters

A	Pointer to the coeff matrix in dBSRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
pc	Pointer to the preconditioning action
itparam	Pointer to parameters for iterative solvers

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou, Xiaozhe Hu

Date

10/26/2010 Modified by Chunsheng Feng on 03/04/2016: add VBiCGstab solver

Definition at line 38 of file itsolver_bsr.c.

10.47.2.2 fasp_solver_dbsr_krylov()

```
INT fasp_solver_dbsr_krylov (
            dBSRmat * A,
            dvector * b,
            dvector * x,
            itsolver_param * itparam )
```

Solve Ax=b by standard Krylov methods for BSR matrices.

Parameters

A	Pointer to the coeff matrix in dBSRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou, Xiaozhe Hu

Date

10/26/2010

Definition at line 131 of file itsolver_bsr.c.

10.47.2.3 fasp_solver_dbsr_krylov_amg()

```
INT fasp_solver_dbsr_krylov_amg (
            dBSRmat * A,
            dvector * b,
            dvector * x,
            itsolver_param * itparam,
            AMG_param * amgparam )
```

Solve Ax=b by AMG preconditioned Krylov methods.

Parameters

A	Pointer to the coeff matrix in dBSRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers
amgparam	Pointer to parameters of AMG

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

03/16/2012

parameters of iterative method

Definition at line 353 of file itsolver_bsr.c.

10.47.2.4 fasp_solver_dbsr_krylov_amg_nk()

```
INT fasp_solver_dbsr_krylov_amg_nk (
            dBSRmat * A,
            dvector * b,
            dvector * x,
            itsolver_param * itparam,
            AMG_param * amgparam,
            dCSRmat * A_nk,
            dCSRmat * P_nk,
            dCSRmat * R_nk )
```

Solve Ax=b by AMG with extra near kernel solve preconditioned Krylov methods.

Parameters

A	Pointer to the coeff matrix in dBSRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers
amgparam	Pointer to parameters of AMG
A_nk	Pointer to the coeff matrix for near kernel space in dBSRmat format
P_nk	Pointer to the prolongation for near kernel space in dBSRmat format
R_nk	Pointer to the restriction for near kernel space in dBSRmat format

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

05/26/2012

Definition at line 495 of file itsolver_bsr.c.

10.47.2.5 fasp_solver_dbsr_krylov_diag()

```
INT fasp_solver_dbsr_krylov_diag (
            dBSRmat * A,
            dvector * b,
            dvector * x,
            itsolver_param * itparam )
```

Solve Ax=b by diagonal preconditioned Krylov methods.

Parameters

A	Pointer to the coeff matrix in dBSRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou, Xiaozhe Hu

Date

10/26/2010

Modified by Chunsheng Feng, Zheng Li on 10/15/2012

Definition at line 182 of file itsolver_bsr.c.

10.47.2.6 fasp_solver_dbsr_krylov_ilu()

```
INT fasp_solver_dbsr_krylov_ilu (
            dBSRmat * A,
            dvector * b,
            dvector * x,
            itsolver_param * itparam,
            ILU_param * iluparam )
```

Solve Ax=b by ILUs preconditioned Krylov methods.

Parameters

A	Pointer to the coeff matrix in dBSRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers
iluparam	Pointer to parameters of ILU

Returns

Iteration number if converges; ERROR otherwise.

Author

Shiquang Zhang, Xiaozhe Hu

Date

10/26/2010

Definition at line 286 of file itsolver_bsr.c.

10.47.2.7 fasp_solver_dbsr_krylov_nk_amg()

```
INT fasp_solver_dbsr_krylov_nk_amg (
            dBSRmat * A,
            dvector * b,
            dvector * x,
            itsolver_param * itparam,
            AMG_param * amgparam,
            const INT nk_dim,
            dvector * nk )
```

Solve Ax=b by AMG preconditioned Krylov methods with extra kernal space.

Parameters

A	Pointer to the coeff matrix in dBSRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers
amgparam	Pointer to parameters of AMG
nk_dim	Dimension of the near kernel spaces
nk	Pointer to the near kernal spaces

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

05/27/2012

parameters of iterative method

Definition at line 654 of file itsolver_bsr.c.

10.48 itsolver_csr.c File Reference

Iterative solvers for dCSRmat matrices.

```
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_dcsr_itsolver (dCSRmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, itsolver_param ∗itparam)

  Solve Ax=b by preconditioned Krylov methods for CSR matrices.
- INT fasp_solver_dcsr_krylov (dCSRmat ∗A, dvector ∗b, dvector ∗x, itsolver_param ∗itparam)

  Solve Ax=b by standard Krylov methods for CSR matrices.
- INT fasp_solver_dcsr_krylov_diag (dCSRmat ∗A, dvector ∗b, dvector ∗x, itsolver_param ∗itparam)

  Solve Ax=b by diagonal preconditioned Krylov methods.
- INT fasp_solver_dcsr_krylov_Schwarz (dCSRmat ∗A, dvector ∗b, dvector ∗x, itsolver_param ∗itparam, Schwarz_param ∗schparam)

  Solve Ax=b by overlapping Schwarz Krylov methods.
- INT fasp_solver_dcsr_krylov_amg (dCSRmat ∗A, dvector ∗b, dvector ∗x, itsolver_param ∗itparam, AMG_param ∗amgparam)

  Solve Ax=b by AMG preconditioned Krylov methods.
- INT fasp_solver_dcsr_krylov_ilu (dCSRmat ∗A, dvector ∗b, dvector ∗x, itsolver_param ∗itparam, ILU_param ∗iluparam)

  Solve Ax=b by ILUs preconditioned Krylov methods.
- INT fasp_solver_dcsr_krylov_ilu_M (dCSRmat ∗A, dvector ∗b, dvector ∗x, itsolver_param ∗itparam, ILU_param ∗iluparam, dCSRmat ∗M)

  Solve Ax=b by ILUs preconditioned Krylov methods: ILU of M as preconditioner.
- INT fasp_solver_dcsr_krylov_amg_nk (dCSRmat ∗A, dvector ∗b, dvector ∗x, itsolver_param ∗itparam, AMG_param ∗amgparam, dCSRmat ∗A_nk, dCSRmat ∗P_nk, dCSRmat ∗R_nk)

  Solve Ax=b by AMG preconditioned Krylov methods with an extra near kernel solve.

10.48.1 Detailed Description

Iterative solvers for dCSRmat matrices.

10.48.2 Function Documentation

10.48.2.1 fasp_solver_dcsr_itsolver()

```
INT fasp_solver_dcsr_itsolver (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            itsolver_param * itparam )
```

Solve Ax=b by preconditioned Krylov methods for CSR matrices.

Parameters

A	Pointer to the coeff matrix in dCSRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
pc	Pointer to the preconditioning action
itparam	Pointer to parameters for iterative solvers

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

09/25/2009

Note

This is an abstract interface for iterative methods. Modified by Chunsheng Feng on 03/04/2016: add VBiCGstab solver

Definition at line 40 of file itsolver_csr.c.

10.48.2.2 fasp_solver_dcsr_krylov()

```
INT fasp_solver_dcsr_krylov (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            itsolver_param * itparam )
```

Solve Ax=b by standard Krylov methods for CSR matrices.

Parameters

A	Pointer to the coeff matrix in dCSRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang, Shiquan Zhang

Date

09/25/2009

Definition at line 149 of file itsolver_csr.c.

10.48.2.3 fasp_solver_dcsr_krylov_amg()

```
INT fasp_solver_dcsr_krylov_amg (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            itsolver_param * itparam,
            AMG_param * amgparam )
```

Solve Ax=b by AMG preconditioned Krylov methods.

Parameters

A	Pointer to the coeff matrix in dCSRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers
amgparam	Pointer to parameters for AMG methods

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

09/25/2009

Definition at line 344 of file itsolver_csr.c.

10.48.2.4 fasp_solver_dcsr_krylov_amg_nk()

```
INT fasp_solver_dcsr_krylov_amg_nk (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            itsolver_param * itparam,
            AMG_param * amgparam,
            dCSRmat * A_nk,
            dCSRmat * P_nk,
            dCSRmat * R_nk )
```

Solve Ax=b by AMG preconditioned Krylov methods with an extra near kernel solve.

Parameters

A	Pointer to the coeff matrix in dCSRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers
amgparam	Pointer to parameters for AMG methods
A_nk	Pointer to the coeff matrix of near kernel space in dCSRmat format
P_nk	Pointer to the prolongation of near kernel space in dCSRmat format
R_nk	Pointer to the restriction of near kernel space in dCSRmat format

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

05/26/2014

Definition at line 617 of file itsolver_csr.c.

10.48.2.5 fasp_solver_dcsr_krylov_diag()

```
INT fasp_solver_dcsr_krylov_diag (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            itsolver_param * itparam )
```

Solve Ax=b by diagonal preconditioned Krylov methods.

Parameters

A	Pointer to the coeff matrix in dCSRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang, Shiquan Zhang

Date

09/25/2009

Definition at line 199 of file itsolver_csr.c.

10.48.2.6 fasp_solver_dcsr_krylov_ilu()

```
INT fasp_solver_dcsr_krylov_ilu (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            itsolver_param * itparam,
            ILU_param * iluparam )
```

Solve Ax=b by ILUs preconditioned Krylov methods.

Parameters

A	Pointer to the coeff matrix in dCSRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers
iluparam	Pointer to parameters for ILU

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang, Shiquan Zhang

Date

09/25/2009

Definition at line 449 of file itsolver_csr.c.

10.48.2.7 fasp_solver_dcsr_krylov_ilu_M()

```
INT fasp_solver_dcsr_krylov_ilu_M (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            itsolver_param * itparam,
            ILU_param * iluparam,
            dCSRmat * M )
```

Solve Ax=b by ILUs preconditioned Krylov methods: ILU of M as preconditioner.

Parameters

A	Pointer to the coeff matrix in dCSRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers
iluparam	Pointer to parameters for ILU
M	Pointer to the preconditioning matrix in dCSRmat format

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

09/25/2009

Note

This function is specially designed for reservoir simulation. Have not been tested in any other places.

Definition at line 533 of file itsolver_csr.c.

10.48.2.8 fasp_solver_dcsr_krylov_Schwarz()

```
INT fasp_solver_dcsr_krylov_Schwarz (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            itsolver_param * itparam,
            Schwarz_param * schparam )
```

Solve Ax=b by overlapping Schwarz Krylov methods.

Parameters

A	Pointer to the coeff matrix in dCSRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers
schparam	Pointer to parameters for Schwarz methods

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

03/21/2011

Modified by Chensong on 07/02/2012: change interface

Definition at line 263 of file itsolver_csr.c.

10.49 itsolver_mf.c File Reference

Iterative solvers using matrix-free spmv operations.

```
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "fasp_block.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_itsolver (mxv_matfree *mf, dvector *b, dvector *x, precond *pc, itsolver_param *itparam)

  Solve Ax=b by preconditioned Krylov methods for CSR matrices.
- INT fasp_solver_krylov (mxv_matfree *mf, dvector *b, dvector *x, itsolver_param *itparam)

  Solve Ax=b by standard Krylov methods – without preconditioner.
- void fasp_solver_itsolver_init (INT matrix_format, mxv_matfree *mf, void *A)

  Initialize itsovlers.

10.49.1 Detailed Description

Iterative solvers using matrix-free spmv operations.

10.49.2 Function Documentation

10.49.2.1 fasp_solver_itsolver()

```
INT fasp_solver_itsolver (
            mxv_matfree * mf,
            dvector * b,
            dvector * x,
            precond * pc,
            itsolver_param * itparam )
```

Solve Ax=b by preconditioned Krylov methods for CSR matrices.

Parameters

mf	Pointer to mxv_matfree matrix-free spmv operation
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
pc	Pointer to the preconditioning action
itparam	Pointer to parameters for iterative solvers

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

09/25/2009

Note

This is an abstract interface for iterative methods.

Modified by Feiteng Huang on 09/19/2012: matrix free

Definition at line 50 of file itsolver_mf.c.

10.49.2.2 fasp_solver_itsolver_init()

```
void fasp_solver_itsolver_init (
            INT matrix_format,
            mxv_matfree * mf,
            void * A )
```

Initialize itsovlers.

Parameters

matrix_format	matrix format
mf	Pointer to mxv_matfree matrix-free spmv operation
A	void pointer to matrix

Author

Feiteng Huang

Date

09/18/2012

Modified by Chensong Zhang on 05/10/2013: Change interface of mat-free mv

Definition at line 197 of file itsolver_mf.c.

10.49.2.3 fasp_solver_krylov()

```
INT fasp_solver_krylov (
            mxv_matfree * mf,
            dvector * b,
            dvector * x,
            itsolver_param * itparam )
```

Solve Ax=b by standard Krylov methods – without preconditioner.

Parameters

mf	Pointer to mxv_matfree matrix-free spmv operation
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers

Returns

Number of iterations if succeed

Author

Chensong Zhang, Shiquan Zhang

Date

09/25/2009

Modified by Feiteng Huang on 09/20/2012: matrix free

Definition at line 150 of file itsolver_mf.c.

10.50 itsolver_str.c File Reference

Iterative solvers for dSTRmat matrices.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_dstr_itsolver (dSTRmat *A, dvector *b, dvector *x, precond *pc, itsolver_param *itparam)

  Solve Ax=b by standard Krylov methods.
- INT fasp_solver_dstr_krylov (dSTRmat *A, dvector *b, dvector *x, itsolver_param *itparam)

  Solve Ax=b by standard Krylov methods.
- INT fasp_solver_dstr_krylov_diag (dSTRmat *A, dvector *b, dvector *x, itsolver_param *itparam)

  Solve Ax=b by diagonal preconditioned Krylov methods.
- INT fasp_solver_dstr_krylov_ilu (dSTRmat *A, dvector *b, dvector *x, itsolver_param *itparam, ILU_param *iluparam)

  Solve Ax=b by structured ILU preconditioned Krylov methods.
- INT fasp_solver_dstr_krylov_blockgs (dSTRmat *A, dvector *b, dvector *x, itsolver_param *itparam, ivector *neigh, ivector *order)

  Solve Ax=b by diagonal preconditioned Krylov methods.

10.50.1 Detailed Description

Iterative solvers for dSTRmat matrices.

10.50.2 Function Documentation

10.50.2.1 fasp_solver_dstr_itsolver()

```
INT fasp_solver_dstr_itsolver (
            dSTRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            itsolver_param * itparam )
```

Solve Ax=b by standard Krylov methods.

Parameters

A	Pointer to the coeff matrix in dSTRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
pc	Pointer to the preconditioning action
itparam	Pointer to parameters for iterative solvers

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

09/25/2009 Modified by Chunsheng Feng on 03/04/2016: add VBiCGstab solver

Definition at line 35 of file itsolver_str.c.

10.50.2.2 fasp_solver_dstr_krylov()

```
INT fasp_solver_dstr_krylov (
            dSTRmat * A,
            dvector * b,
            dvector * x,
            itsolver_param * itparam )
```

Solve Ax=b by standard Krylov methods.

Parameters

A	Pointer to the coeff matrix in dSTRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou

Date

04/25/2010

Definition at line 123 of file itsolver_str.c.

10.50.2.3 fasp_solver_dstr_krylov_blockgs()

```
INT fasp_solver_dstr_krylov_blockgs (
            dSTRmat * A,
            dvector * b,
            dvector * x,
            itsolver_param * itparam,
            ivector * neigh,
            ivector * order )
```

Solve Ax=b by diagonal preconditioned Krylov methods.

Parameters

A	Pointer to the coeff matrix in dSTRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers
neigh	Pointer to neighbor vector
order	Pointer to solver ordering

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

10/10/2010

Definition at line 330 of file itsolver_str.c.

10.50.2.4 fasp_solver_dstr_krylov_diag()

```
INT fasp_solver_dstr_krylov_diag (
            dSTRmat * A,
            dvector * b,
            dvector * x,
            itsolver_param * itparam )
```

Solve Ax=b by diagonal preconditioned Krylov methods.

Parameters

A	Pointer to the coeff matrix in dSTRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou

Date

4/23/2010

Definition at line 171 of file itsolver_str.c.

10.50.2.5 fasp_solver_dstr_krylov_ilu()

```
INT fasp_solver_dstr_krylov_ilu (
            dSTRmat * A,
            dvector * b,
            dvector * x,
            itsolver_param * itparam,
            ILU_param * iluparam )
```

Solve Ax=b by structured ILU preconditioned Krylov methods.

Parameters

A	Pointer to the coeff matrix in dSTRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers
iluparam	Pointer to parameters for ILU

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

05/01/2010

Definition at line 237 of file itsolver_str.c.

10.51 lu.c File Reference

LU decomposition and direct solver for small dense matrices.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- SHORT fasp_smat_lu_decomp (REAL *A, INT pivot[ ], const INT n)

  LU decomposition of A usind Doolittle's method.
- SHORT fasp_smat_lu_solve (REAL *A, REAL b[ ], INT pivot[ ], REAL x[ ], const INT n)

  Solving Ax=b using LU decomposition.

10.51.1 Detailed Description

LU decomposition and direct solver for small dense matrices.

10.51.2 Function Documentation

10.51.2.1 fasp_smat_lu_decomp()

```
SHORT fasp_smat_lu_decomp (
            REAL * A,
            INT pivot[ ],
            const INT n )
```

LU decomposition of A usind Doolittle's method.

Parameters

A	Pointer to the full matrix
pivot	Pivoting positions
n	Size of matrix A

Returns

FASP_SUCCESS if successed; otherwise, error information.

Note

Use Doolittle's method to decompose the n x n matrix A into a unit lower triangular matrix L and an upper triangular matrix U such that A = LU. The matrices L and U replace the matrix A. The diagonal elements of L are 1 and are not stored.
The Doolittle method with partial pivoting is: Determine the pivot row and interchange the current row with the pivot row, then assuming that row k is the current row, k = 0, ..., n - 1 evaluate in order the following pair of expressions U[k][j] = A[k][j] - (L[k][0]*U[0][j] + ... + L[k][k-1]*U[k-1][j]) for j = k, k+1, ... , n-1 L[i][k] = (A[i][k] - (L[i][0]*U[0][k] + . + L[i][k-1]*U[k-1][k])) / U[k][k] for i = k+1, ... , n-1.

Author

Xuehai Huang

Date

04/02/2009

Definition at line 46 of file lu.c.

10.51.2.2 fasp_smat_lu_solve()

```
SHORT fasp_smat_lu_solve (
            REAL * A,
            REAL b[ ],
            INT pivot[ ],
            REAL x[ ],
            const INT n )
```

Solving Ax=b using LU decomposition.

Parameters

A	Pointer to the full matrix
b	Right hand side array
pivot	Pivoting positions
x	Pointer to the solution array
n	Size of matrix A

Returns

FASP_SUCCESS if successed; otherwise, error information.

Note

This routine uses Doolittle's method to solve the linear equation Ax = b. This routine is called after the matrix A has been decomposed into a product of a unit lower triangular matrix L and an upper triangular matrix U with pivoting. The solution proceeds by solving the linear equation Ly = b for y and subsequently solving the linear equation Ux = y for x.

Author

Xuehai Huang

Date

04/02/2009

Definition at line 117 of file lu.c.

10.52 memory.c File Reference

Memory allocation and deallocation subroutines.

```
#include "fasp.h"
```

Functions

- void ∗ fasp_mem_calloc (LONGLONG size, INT type)

  1M = 1024∗1024
- void ∗ fasp_mem_realloc (void ∗oldmem, LONGLONG tsize)

  Reallocate, initiate, and check memory.
- void fasp_mem_free (void ∗mem)

  Free up previous allocated memory body.
- void fasp_mem_usage ()

  Show total allocated memory currently.
- SHORT fasp_mem_check (void ∗ptr, const char ∗message, INT ERR)

  Check wether a point is null or not.
- SHORT fasp_mem_iludata_check (ILU_data ∗iludata)

  Check wether a ILU_data has enough work space.
- SHORT fasp_mem_dcsr_check (dCSRmat ∗A)

  Check wether a dCSRmat A has sucessfully allocated memory.

Variables

- unsigned INT total_alloc_mem = 0
- unsigned INT total_alloc_count = 0

  Total allocated memory amount.
- const INT Million = 1048576

  Total number of allocations.

10.52.1 Detailed Description

Memory allocation and deallocation subroutines.

10.52.2 Function Documentation

10.52.2.1 fasp_mem_calloc()

```
void * fasp_mem_calloc (
            LONGLONG size,
            INT type )
```

1M = 1024*1024

Allocate, initiate, and check memory

Parameters

size	Number of memory blocks
type	Size of memory blocks

Returns

Void pointer to the allocated memory

Author

Chensong Zhang

Date

2010/08/12

Modified by Chunsheng Feng on 12/20/2013 Modified by Chunsheng Feng on 07/23/2013 Modified by Chunsheng Feng on 07/30/2013 Modified by Chensong Zhang on 07/30/2013: print error if failed

Definition at line 58 of file memory.c.

10.52.2.2 fasp_mem_check()

```
SHORT fasp_mem_check (
            void * ptr,
            const char * message,
            INT ERR )
```

Check wether a point is null or not.

Parameters

ptr	Void pointer to be checked
message	Error message to print
ERR	Integer error code

Returns

FASP_SUCCESS or error code

Author

Chensong Zhang

Date

11/16/2009

Definition at line 195 of file memory.c.

10.52.2.3 fasp_mem_dcsr_check()

```
SHORT fasp_mem_dcsr_check (
            dCSRmat * A )
```

Check wether a dCSRmat A has sucessfully allocated memory.

Parameters

A	Pointer to be cheked

Returns

FASP_SUCCESS if success, else ERROR message (negative value)

Author

Xiaozhe Hu

Date

11/27/09

Definition at line 246 of file memory.c.

10.52.2.4 fasp_mem_free()

```
void fasp_mem_free (
            void * mem )
```

Free up previous allocated memory body.

Parameters

mem	Pointer to the memory body need to be freed

Returns

NULL pointer

Author

Chensong Zhang

Date

2010/12/24

Definition at line 148 of file memory.c.

10.52.2.5 fasp_mem_iludata_check()

```
SHORT fasp_mem_iludata_check (
            ILU_data * iludata )
```

Check wether a ILU_data has enough work space.

Parameters

iludata	Pointer to be cheked

Returns

FASP_SUCCESS if success, else ERROR (negative value)

Author

Xiaozhe Hu, Chensong Zhang

Date

11/27/09

Definition at line 220 of file memory.c.

10.52.2.6 fasp_mem_realloc()

```
void * fasp_mem_realloc (
            void * oldmem,
            LONGLONG type )
```

Reallocate, initiate, and check memory.

Parameters

oldmem	Pointer to the existing mem block
type	Size of memory blocks

Returns

Void pointer to the reallocated memory

Author

Chensong Zhang

Date

2010/08/12

Modified by Chunsheng Feng on 07/23/2013 Modified by Chensong Zhang on 07/30/2013: print error if failed

Definition at line 108 of file memory.c.

10.52.2.7 fasp_mem_usage()

```
void fasp_mem_usage ( )
```

Show total allocated memory currently.

Author

Chensong Zhang

Date

2010/08/12

Definition at line 173 of file memory.c.

10.52.3 Variable Documentation

10.52.3.1 total_alloc_count

```
unsigned INT total_alloc_count = 0
```

Total allocated memory amount.

total allocation times

Definition at line 33 of file memory.c.

10.52.3.2 total_alloc_mem

```
unsigned INT total_alloc_mem = 0
```

total allocated memory

Definition at line 32 of file memory.c.

10.53 message.c File Reference

Output some useful messages.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void print_itinfo (const INT ptrlvl, const INT stop_type, const INT iter, const REAL relres, const REAL absres, const REAL factor)

  Print out iteration information for iterative solvers.
- void print_amgcomplexity (AMG_data ∗mgl, const SHORT prtlvl)

  Print complexities of AMG method.
- void print_amgcomplexity_bsr (AMG_data_bsr ∗mgl, const SHORT prtlvl)

  Print complexities of AMG method for BSR matrices.
- void print_cputime (const char ∗message, const REAL cputime)

  Print CPU walltime.
- void print_message (const INT ptrlvl, const char ∗message)

  Print output information if necessary.
- void fasp_chkerr (const SHORT status, const char ∗fctname)

  Check error status and print out error messages before quit.

10.53.1 Detailed Description

Output some useful messages.

Note

These routines are meant for internal use only.

10.53.2 Function Documentation

10.53.2.1 fasp_chkerr()

```
void fasp_chkerr (
            const SHORT status,
            const char * fctname )
```

Check error status and print out error messages before quit.

Parameters

status	Error status
fctname	Function name where this routine is called

Author

Chensong Zhang

Date

01/10/2012

Definition at line 199 of file message.c.

10.53.2.2 print_amgcomplexity()

```
void void print_amgcomplexity (
            AMG_data * mgl,
            const SHORT prtlvl )
```

Print complexities of AMG method.

Parameters

mgl	Multilevel hierachy for AMG
prtlvl	How much information to print

Author

Chensong Zhang

Date

11/16/2009

Definition at line 79 of file message.c.

10.53.2.3 print_amgcomplexity_bsr()

```
void void print_amgcomplexity_bsr (
            AMG_data_bsr * mgl,
            const SHORT prtlvl )
```

Print complexities of AMG method for BSR matrices.

Parameters

mgl	Multilevel hierachy for AMG
prtlvl	How much information to print

Author

Chensong Zhang

Date

05/10/2013

Definition at line 122 of file message.c.

10.53.2.4 print_cputime()

```
void void print_cputime (
            const char * message,
            const REAL cputime )
```

Print CPU walltime.

Parameters

message	Some string to print out
cputime	Walltime since start to end

Author

Chensong Zhang

Date

04/10/2012

Definition at line 165 of file message.c.

10.53.2.5 print_itinfo()

```
void print_itinfo (
            const INT ptrlvl,
            const INT stop_type,
            const INT iter,
            const REAL relres,
            const REAL absres,
            const REAL factor )
```

Print out iteration information for iterative solvers.

Parameters

ptrlvl	Level for output
stop_type	Type of stopping criteria
iter	Number of iterations
relres	Relative residual of different kinds
absres	Absolute residual of different kinds
factor	Contraction factor

Author

Chensong Zhang

Date

11/16/2009

Modified by Chensong Zhang on 03/28/2013: Output initial guess Modified by Chensong Zhang on 04/05/2013: Fix a typo

Definition at line 36 of file message.c.

10.53.2.6 print_message()

```
void print_message (
            const INT ptrlvl,
            const char * message )
```

Print output information if necessary.

Parameters

ptrlvl	Level for output
message	Error message to print

Author

Chensong Zhang

Date

11/16/2009

Definition at line 182 of file message.c.

10.54 mgcycle.c File Reference

Abstract multigrid cycle – non-recursive version.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "mg_util.inl"
```

Functions

- void fasp_solver_mgcycle (AMG_data *mgl, AMG_param *param)

  Solve Ax=b with non-recursive multigrid cycle.
- void fasp_solver_mgcycle_bsr (AMG_data_bsr *mgl, AMG_param *param)

  Solve Ax=b with non-recursive multigrid cycle.

10.54.1 Detailed Description

Abstract multigrid cycle – non-recursive version.

10.54.2 Function Documentation

10.54.2.1 fasp_solver_mgcycle()

```
void fasp_solver_mgcycle (
            AMG_data * mgl,
            AMG_param * param )
```

Solve Ax=b with non-recursive multigrid cycle.

Parameters

mgl	Pointer to AMG data: AMG_data
param	Pointer to AMG parameters: AMG_param

Author

Chensong Zhang

Date

10/06/2010

Modified by Chensong Zhang on 12/13/2011 Modified by Chensong Zhang on 02/27/2013: update direct solvers. Modified by Chensong Zhang on 12/30/2014: update Schwarz smoothers.

Definition at line 40 of file mgcycle.c.

10.54.2.2 fasp_solver_mgcycle_bsr()

```
void fasp_solver_mgcycle_bsr (
            AMG_data_bsr * mgl,
            AMG_param * param )
```

Solve Ax=b with non-recursive multigrid cycle.

Parameters

mgl	Pointer to AMG data: AMG_data_bsr
param	Pointer to AMG parameters: AMG_param

Author

Xiaozhe Hu

Date

08/07/2011

Definition at line 264 of file mgcycle.c.

10.55 mgrecur.c File Reference

Abstract multigrid cycle – recursive version.

```
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "mg_util.inl"
```

Functions

- void fasp_solver_mgrecur (AMG_data *mgl, AMG_param *param, INT level)
  Solve Ax=b with recursive multigrid K-cycle.

10.55.1 Detailed Description

Abstract multigrid cycle – recursive version.

Note

Not used any more. Will be removed! –Chensong

10.55.2 Function Documentation

10.55.2.1 fasp_solver_mgrecur()

```
void fasp_solver_mgrecur (
            AMG_data * mgl,
            AMG_param * param,
            INT level )
```

Solve Ax=b with recursive multigrid K-cycle.

Parameters

mgl	Pointer to AMG data: AMG_data
param	Pointer to AMG parameters: AMG_param
level	Index of the current level

Author

Xuehai Huang, Chensong Zhang

Date

04/06/2010

Modified by Chensong Zhang on 01/10/2012 Modified by Chensong Zhang on 02/27/2013: update direct solvers.

Definition at line 33 of file mgrecur.c.

10.56 ordering.c File Reference

Subroutines for ordering, merging, removing duplicated integers.

```
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- INT fasp_BinarySearch (INT ∗list, const INT value, const INT nlist)
  Binary Search.
- INT fasp_aux_unique (INT numbers[ ], const INT size)
  Remove duplicates in an sorted (ascending order) array.
- void fasp_aux_merge (INT numbers[ ], INT work[ ], INT left, INT mid, INT right)
  Merge two sorted arrays.
- void fasp_aux_msort (INT numbers[ ], INT work[ ], INT left, INT right)
  Sort the INT array in ascending order with the merge sort algorithm.
- void fasp_aux_iQuickSort (INT ∗a, INT left, INT right)
  Sort the array (INT type) in ascending order with the quick sorting algorithm.
- void fasp_aux_dQuickSort (REAL ∗a, INT left, INT right)
  Sort the array (REAL type) in ascending order with the quick sorting algorithm.
- void fasp_aux_iQuickSortIndex (INT ∗a, INT left, INT right, INT ∗index)
  Reorder the index of (INT type) so that 'a' is in ascending order.
- void fasp_aux_dQuickSortIndex (REAL ∗a, INT left, INT right, INT ∗index)
  Reorder the index of (REAL type) so that 'a' is ascending in such order.
- void fasp_dcsr_CMK_order (const dCSRmat ∗A, INT ∗order, INT ∗oindex)
  Ordering vertices of matrix graph corresponding to A.
- void fasp_dcsr_RCMK_order (const dCSRmat ∗A, INT ∗order, INT ∗oindex, INT ∗rorder)
  Resverse CMK ordering.
- void fasp_topological_sorting_ilu (ILU_data ∗iludata)
  Reordering vertices according to level schedule strategy.
- void fasp_multicolors_independent_set (AMG_data ∗mgl, INT gslvl)
  Coloring vertices of adjacency graph of A.

10.56.1 Detailed Description

Subroutines for ordering, merging, removing duplicated integers.

10.56.2 Function Documentation

10.56.2.1 fasp_aux_dQuickSort()

```
void fasp_aux_dQuickSort (
            REAL * a,
            INT left,
            INT right )
```

Sort the array (REAL type) in ascending order with the quick sorting algorithm.

Parameters

a	Pointer to the array needed to be sorted
left	Starting index
right	Ending index

Author

Zhiyang Zhou

Date

2009/11/28

Note

'left' and 'right' are usually set to be 0 and n-1, respectively where n is the length of 'a'.

Definition at line 241 of file ordering.c.

10.56.2.2 fasp_aux_dQuickSortIndex()

```
void fasp_aux_dQuickSortIndex (
            REAL * a,
            INT left,
            INT right,
            INT * index )
```

Reorder the index of (REAL type) so that 'a' is ascending in such order.

Parameters

a	Pointer to the array
left	Starting index
right	Ending index
index	Index of 'a' (out)

Author

Zhiyang Zhou

Date

2009/12/02

Note

'left' and 'right' are usually set to be 0 and n-1,respectively,where n is the length of 'a'. 'index' should be initialized in the nature order and it has the same length as 'a'.

Definition at line 322 of file ordering.c.

10.56.2.3 fasp_aux_iQuickSort()

```
void fasp_aux_iQuickSort (
            INT * a,
            INT left,
            INT right )
```

Sort the array (INT type) in ascending order with the quick sorting algorithm.

Parameters

a	Pointer to the array needed to be sorted
left	Starting index
right	Ending index

Author

Zhiyang Zhou

Date

11/28/2009

Note

'left' and 'right' are usually set to be 0 and n-1, respectively where n is the length of 'a'.

Definition at line 203 of file ordering.c.

10.56.2.4 fasp_aux_iQuickSortIndex()

```
void fasp_aux_iQuickSortIndex (
            INT * a,
            INT left,
            INT right,
            INT * index )
```

Reorder the index of (INT type) so that 'a' is in ascending order.

Parameters

a	Pointer to the array
left	Starting index
right	Ending index
index	Index of 'a' (out)

Author

Zhiyang Zhou

Date

2009/12/02

Note

'left' and 'right' are usually set to be 0 and n-1,respectively,where n is the length of 'a'. 'index' should be initialized in the nature order and it has the same length as 'a'.

Definition at line 281 of file ordering.c.

10.56.2.5 fasp_aux_merge()

```
void fasp_aux_merge (
            INT numbers[],
            INT work[],
            INT left,
            INT mid,
            INT right )
```

Merge two sorted arrays.

Parameters

numbers	Pointer to the array needed to be sorted
work	Pointer to the work array with same size as numbers
left	Starting index of array 1
mid	Starting index of array 2
right	Ending index of array 1 and 2

Author

Chensong Zhang

Date

11/21/2010

Note

Both arrays are stored in numbers! Arrays should be pre-sorted!

Definition at line 110 of file ordering.c.

10.56.2.6 fasp_aux_msort()

```
void fasp_aux_msort (
            INT numbers[],
            INT work[],
            INT left,
            INT right )
```

Sort the INT array in ascending order with the merge sort algorithm.

Parameters

numbers	Pointer to the array needed to be sorted
work	Pointer to the work array with same size as numbers
left	Starting index
right	Ending index

Author

Chensong Zhang

Date

11/21/2010

Note

'left' and 'right' are usually set to be 0 and n-1, respectively

Definition at line 172 of file ordering.c.

10.56.2.7 fasp_aux_unique()

```
INT fasp_aux_unique (
            INT numbers[],
            const INT size )
```

Remove duplicates in an sorted (ascending order) array.

Parameters

numbers	Pointer to the array needed to be sorted (in/out)
size	Length of the target array

Returns

New size after removing duplicates

Author

Chensong Zhang

Date

11/21/2010

Note

Operation is in place. Does not use any extra or temporary storage.

Definition at line 77 of file ordering.c.

10.56.2.8 fasp_BinarySearch()

```
INT fasp_BinarySearch (
            INT * list,
            const INT value,
            const INT nlist )
```

Binary Search.

Parameters

list	Pointer to a set of values
value	The target
nlist	Length of the array list

Returns

The location of value in array list if succeeded; otherwise, return -1.

Author

Chunsheng Feng

Date

03/01/2011

Definition at line 32 of file ordering.c.

10.56.2.9 fasp_dcsr_CMK_order()

```
void fasp_dcsr_CMK_order (
            const dCSRmat * A,
            INT * order,
            INT * oindex )
```

Ordering vertices of matrix graph corresponding to A.

Parameters

A	Pointer to matrix
oindex	Pointer to index of vertices in order
order	Pointer to vertices with increasing degree

Author

Zheng Li, Chensong Zhang

Date

05/28/2014

Definition at line 358 of file ordering.c.

10.56.2.10 fasp_dcsr_RCMK_order()

```
void fasp_dcsr_RCMK_order (
            const dCSRmat * A,
            INT * order,
            INT * oindex,
            INT * rorder )
```

Resverse CMK ordering.

Parameters

A	Pointer to matrix
order	Pointer to vertices with increasing degree
oindex	Pointer to index of vertices in order
rorder	Pointer to reverse order

Author

Zheng Li, Chensong Zhang

Date

10/10/2014

Definition at line 407 of file ordering.c.

10.56.2.11 fasp_multicolors_independent_set()

```
void fasp_multicolors_independent_set (
            AMG_data * mgl,
            INT gslvl )
```

Coloring vertices of adjacency graph of A.

Parameters

mgl	Pointer to input matrix
gslvl	Used to specify levels of AMG using multicolor smoothing

Author

Zheng Li, Chunsheng Feng

Date

12/04/2016

Definition at line 514 of file ordering.c.

10.56.2.12 fasp_topological_sorting_ilu()

```
void fasp_topological_sorting_ilu (
            ILU_data * iludata )
```

Reordering vertices according to level schedule strategy.

Parameters

iludata	Pointer to iludata

Author

Zheng Li, Chensong Zhang

Date

12/04/2016

Definition at line 432 of file ordering.c.

10.57 parameters.c File Reference

Initialize, set, or print input data and parameters.

```
#include <stdio.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_param_set (int argc, const char ∗argv[ ], input_param ∗iniparam)

    Read input from command-line arguments.
- void fasp_param_init (input_param ∗iniparam, itsolver_param ∗itsparam, AMG_param ∗amgparam, ILU_param ∗iluparam, Schwarz_param ∗schparam)

    Initialize parameters, global variables, etc.
- void fasp_param_input_init (input_param ∗iniparam)

    Initialize input parameters.
- void fasp_param_amg_init (AMG_param ∗amgparam)

    Initialize AMG parameters.
- void fasp_param_solver_init (itsolver_param ∗itsparam)

    Initialize itsolver_param.
- void fasp_param_ilu_init (ILU_param ∗iluparam)

    Initialize ILU parameters.
- void fasp_param_Schwarz_init (Schwarz_param ∗schparam)

    Initialize Schwarz parameters.
- void fasp_param_amg_set (AMG_param ∗param, input_param ∗iniparam)

    Set AMG_param from INPUT.
- void fasp_param_ilu_set (ILU_param ∗iluparam, input_param ∗iniparam)

    Set ILU_param with INPUT.
- void fasp_param_Schwarz_set (Schwarz_param ∗schparam, input_param ∗iniparam)

    Set Schwarz_param with INPUT.
- void fasp_param_solver_set (itsolver_param ∗itsparam, input_param ∗iniparam)

    Set itsolver_param with INPUT.
- void fasp_param_amg_to_prec (precond_data ∗pcdata, AMG_param ∗amgparam)

    Set precond_data with AMG_param.

- void fasp_param_prec_to_amg (AMG_param *amgparam, precond_data *pcdata)

  Set AMG_param with precond_data.
- void fasp_param_amg_to_prec_bsr (precond_data_bsr *pcdata, AMG_param *amgparam)

  Set precond_data_bsr with AMG_param.
- void fasp_param_prec_to_amg_bsr (AMG_param *amgparam, precond_data_bsr *pcdata)

  Set AMG_param with precond_data.
- void fasp_param_amg_print (AMG_param *param)

  Print out AMG parameters.
- void fasp_param_ilu_print (ILU_param *param)

  Print out ILU parameters.
- void fasp_param_Schwarz_print (Schwarz_param *param)

  Print out Schwarz parameters.
- void fasp_param_solver_print (itsolver_param *param)

  Print out itsolver parameters.

10.57.1 Detailed Description

Initialize, set, or print input data and parameters.

10.57.2 Function Documentation

10.57.2.1 fasp_param_amg_init()

```
void fasp_param_amg_init (
            AMG_param * amgparam )
```

Initialize AMG parameters.

Parameters

amgparam	Parameters for AMG

Author

Chensong Zhang

Date

2010/04/03

Definition at line 390 of file parameters.c.

10.57.2.2 fasp_param_amg_print()

```
void fasp_param_amg_print (
            AMG_param * param )
```

Print out AMG parameters.

Parameters

param	Parameters for AMG

Author

Chensong Zhang

Date

2010/03/22

Definition at line 797 of file parameters.c.

10.57.2.3 fasp_param_amg_set()

```
void fasp_param_amg_set (
            AMG_param * param,
            input_param * iniparam )
```

Set AMG_param from INPUT.

Parameters

param	Parameters for AMG
iniparam	Input parameters

Author

Chensong Zhang

Date

2010/03/23

Definition at line 518 of file parameters.c.

10.57.2.4 fasp_param_amg_to_prec()

```
void fasp_param_amg_to_prec (
            precond_data * pcdata,
            AMG_param * amgparam )
```

Set precond_data with AMG_param.

Parameters

pcdata	Preconditioning data structure
amgparam	Parameters for AMG

Author

Chensong Zhang

Date

2011/01/10

Definition at line 666 of file parameters.c.

10.57.2.5 fasp_param_amg_to_prec_bsr()

```
void fasp_param_amg_to_prec_bsr (
            precond_data_bsr * pcdata,
            AMG_param * amgparam )
```

Set precond_data_bsr with AMG_param.

Parameters

pcdata	Preconditioning data structure
amgparam	Parameters for AMG

Author

Xiaozhe Hu

Date

02/06/2012

Definition at line 733 of file parameters.c.

10.57.2.6 fasp_param_ilu_init()

```
void fasp_param_ilu_init (
            ILU_param * iluparam )
```

Initialize ILU parameters.

Parameters

iluparam	Parameters for ILU

Author

Chensong Zhang

Date

2010/04/06

Definition at line 476 of file parameters.c.

10.57.2.7 fasp_param_ilu_print()

```
void fasp_param_ilu_print (
            ILU_param * param )
```

Print out ILU parameters.

Parameters

param	Parameters for ILU

Author

Chensong Zhang

Date

2011/12/20

Definition at line 898 of file parameters.c.

10.57.2.8 fasp_param_ilu_set()

```
void fasp_param_ilu_set (
            ILU_param * iluparam,
            input_param * iniparam )
```

Set ILU_param with INPUT.

Parameters

iluparam	Parameters for ILU
iniparam	Input parameters

Author

Chensong Zhang

Date

2010/04/03

Definition at line 593 of file parameters.c.

10.57.2.9 fasp_param_init()

```
void fasp_param_init (
            input_param * iniparam,
            itsolver_param * itsparam,
            AMG_param * amgparam,
            ILU_param * iluparam,
            Schwarz_param * schparam )
```

Initialize parameters, global variables, etc.

Parameters

iniparam	Input parameters
itsparam	Iterative solver parameters
amgparam	AMG parameters
iluparam	ILU parameters
schparam	Schwarz parameters

Author

Chensong Zhang

Date

2010/08/12

Modified by Xiaozhe Hu (01/23/2011): initialize, then set value Modified by Chensong Zhang (09/12/2012): find a bug during debugging in VS08 Modified by Chensong Zhang (12/29/2013): rewritten

Definition at line 270 of file parameters.c.

10.57.2.10 fasp_param_input_init()

```
void fasp_param_input_init (
            input_param * iniparam )
```

Initialize input parameters.

Parameters

iniparam	Input parameters

Author

Chensong Zhang

Date

2010/03/20

Definition at line 310 of file parameters.c.

10.57.2.11 fasp_param_prec_to_amg()

```
void fasp_param_prec_to_amg (
            AMG_param * amgparam,
            precond_data * pcdata )
```

Set AMG_param with precond_data.

Parameters

amgparam	Parameters for AMG
pcdata	Preconditioning data structure

Author

Chensong Zhang

Date

2011/01/10

Definition at line 701 of file parameters.c.

10.57.2.12 fasp_param_prec_to_amg_bsr()

```
void fasp_param_prec_to_amg_bsr (
            AMG_param * amgparam,
            precond_data_bsr * pcdata )
```

Set AMG_param with precond_data.

Parameters

amgparam	Parameters for AMG
pcdata	Preconditioning data structure

Author

Xiaozhe Hu

Date

02/06/2012

Definition at line 767 of file parameters.c.

10.57.2.13 fasp_param_Schwarz_init()

```
void fasp_param_Schwarz_init (
            Schwarz_param * schparam )
```

Initialize Schwarz parameters.

Parameters

schparam	Parameters for Schwarz method

Author

Xiaozhe Hu

Date

05/22/2012

Modified by Chensong Zhang on 10/10/2014: Add block solver type

Definition at line 498 of file parameters.c.

10.57.2.14 fasp_param_Schwarz_print()

```
void fasp_param_Schwarz_print (
            Schwarz_param * param )
```

Print out Schwarz parameters.

Parameters

param	Parameters for Schwarz

Author

Xiaozhe Hu

Date

05/22/2012

Definition at line 928 of file parameters.c.

10.57.2.15 fasp_param_Schwarz_set()

```
void fasp_param_Schwarz_set (
            Schwarz_param * schparam,
            input_param * iniparam )
```

Set Schwarz_param with INPUT.

Parameters

schparam	Parameters for Schwarz method
iniparam	Input parameters

Author

Xiaozhe Hu

Date

05/22/2012

Definition at line 615 of file parameters.c.

10.57.2.16 fasp_param_set()

```
void fasp_param_set (
            int argc,
            const char * argv[],
            input_param * iniparam )
```

Read input from command-line arguments.

Parameters

argc	Number of arg input
argv	Input arguments
iniparam	Parameters to be set

Author

Chensong Zhang

Date

12/29/2013

Definition at line 27 of file parameters.c.

10.57.2.17 fasp_param_solver_init()

```
void fasp_param_solver_init (
            itsolver_param * itsparam )
```

Initialize itsolver_param.

Parameters

itsparam	Parameters for iterative solvers

Author

Chensong Zhang

Date

2010/03/23

Definition at line 455 of file parameters.c.

10.57.2.18 fasp_param_solver_print()

```
void fasp_param_solver_print (
            itsolver_param * param )
```

Print out itsolver parameters.

Parameters

param	Paramters for iterative solvers

Author

Chensong Zhang

Date

2011/12/20

Definition at line 957 of file parameters.c.

10.57.2.19 fasp_param_solver_set()

```
void fasp_param_solver_set (
            itsolver_param * itsparam,
            input_param * iniparam )
```

Set itsolver_param with INPUT.

Parameters

itsparam	Parameters for iterative solvers
iniparam	Input parameters

Author

Chensong Zhang

Date

2010/03/23

Definition at line 636 of file parameters.c.

10.58 pbcgs.c File Reference

Krylov subspace methods – Preconditioned BiCGstab.

```
#include <math.h>
#include <float.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_dcsr_pbcgs (dCSRmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned BiCGstab method for solving Au=b.
- INT fasp_solver_dcsr_pvbcgs (dCSRmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned BiCGstab method for solving Au=b, Rewritten from Matlab 2011a.
- INT fasp_solver_dbsr_pbcgs (dBSRmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned BiCGstab method for solving Au=b.
- INT fasp_solver_dbsr_pvbcgs (dBSRmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned BiCGstab method for solving Au=b, Rewritten from Matlab 2011a.
- INT fasp_solver_dblc_pbcgs (dBLCmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  A preconditioned BiCGstab method for solving Au=b.
- INT fasp_solver_dblc_pvbcgs (dBLCmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned BiCGstab method for solving Au=b, Rewritten from Matlab 2011a.
- INT fasp_solver_dstr_pbcgs (dSTRmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned BiCGstab method for solving Au=b.
- INT fasp_solver_dstr_pvbcgs (dSTRmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned BiCGstab method for solving Au=b, Rewritten from Matlab 2011a.

10.58.1 Detailed Description

Krylov subspace methods – Preconditioned BiCGstab.

Abstract algorithm

PBICGStab method to solve A∗x=b is to generate {x_k} to approximate x

Note: We generate a series of {p_k} such that V_k=span{p_1,...,p_k}.

Step 0. Given A, b, x_0, M

Step 1. Compute residual r_0 = b-A∗x_0 and convergence check;

Step 2. Initialization z_0 = M^{-1}∗r_0, p_0=z_0;

Step 3. Main loop ...

FOR k = 0:MaxIt

- get step size alpha = f(r_k,z_k,p_k);
- update solution: x_{k+1} = x_k + alpha∗p_k;
- perform stagnation check;
- update residual: r_{k+1} = r_k - alpha∗(A∗p_k);
- perform residual check;
- obtain p_{k+1} using {p_0, p_1, ... , p_k};
- prepare for next iteration;
- print the result of k-th iteration; END FOR

Convergence check: norm(r)/norm(b) < tol

Stagnation check:

- IF norm(alpha∗p_k)/norm(x_{k+1}) < tol_stag
    1. compute r=b-A∗x_{k+1};
    2. convergence check;
    3. IF (not converged & restart_number < Max_Stag_Check) restart;
- END IF

Residual check:

- IF norm(r_{k+1})/norm(b) < tol
    1. compute the real residual r = b-A∗x_{k+1};
    2. convergence check;
    3. IF (not converged & restart_number < Max_Res_Check) restart;
- END IF

Note

Refer to Y. Saad 2003 Iterative methods for sparse linear systems (2nd Edition), SIAM
See spbcgs.c for a safer version

10.58.2 Function Documentation

10.58.2.1 fasp_solver_dblc_pbcgs()

```
INT fasp_solver_dblc_pbcgs (
            dBLCmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

A preconditioned BiCGstab method for solving Au=b.

Parameters

A	Pointer to the coefficient matrix
b	Pointer to the dvector of right hand side
u	Pointer to the dvector of DOFs
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

05/24/2010

Rewritten by Chensong Zhang on 04/30/2012 Modified by Feiteng Huang on 06/01/2012: fix restart param-init Modified by Chensong Zhang on 03/31/2013

Definition at line 1452 of file pbcgs.c.

10.58.2.2 fasp_solver_dblc_pvbcgs()

```
INT fasp_solver_dblc_pvbcgs (
            dBLCmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned BiCGstab method for solving Au=b, Rewritten from Matlab 2011a.

Parameters

A	Pointer to the coefficient matrix
b	Pointer to the dvector of right hand side
u	Pointer to the dvector of DOFs
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chunsheng Feng

Date

03/04/2016

Definition at line 1792 of file pbcgs.c.

10.58.2.3 fasp_solver_dbsr_pbcgs()

```
INT fasp_solver_dbsr_pbcgs (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned BiCGstab method for solving Au=b.

Parameters

A	Pointer to the coefficient matrix
b	Pointer to the dvector of right hand side
u	Pointer to the dvector of DOFs
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

09/09/2009

Rewritten by Chensong Zhang on 04/30/2012 Modified by Feiteng Huang on 06/01/2012: fix restart param-init Modified by Chensong Zhang on 03/31/2013

Definition at line 771 of file pbcgs.c.

10.58.2.4 fasp_solver_dbsr_pvbcgs()

```
INT fasp_solver_dbsr_pvbcgs (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned BiCGstab method for solving Au=b, Rewritten from Matlab 2011a.

Parameters

A	Pointer to the coefficient matrix
b	Pointer to the dvector of right hand side
u	Pointer to the dvector of DOFs
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chunsheng Feng

Date

03/04/2016

Definition at line 1111 of file pbcgs.c.

10.58.2.5 fasp_solver_dcsr_pbcgs()

```
INT fasp_solver_dcsr_pbcgs (
            dCSRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned BiCGstab method for solving Au=b.

Parameters

A	Pointer to the coefficient matrix
b	Pointer to the dvector of right hand side
u	Pointer to the dvector of DOFs
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

09/09/2009

Rewritten by Chensong Zhang on 04/30/2012 Modified by Feiteng Huang on 06/01/2012: fix restart param-init Modified by Chensong Zhang on 03/31/2013

Definition at line 89 of file pbcgs.c.

10.58.2.6 fasp_solver_dcsr_pvbcgs()

```
INT fasp_solver_dcsr_pvbcgs (
            dCSRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned BiCGstab method for solving Au=b, Rewritten from Matlab 2011a.

Parameters

A	Pointer to the coefficient matrix
b	Pointer to the dvector of right hand side
u	Pointer to the dvector of DOFs
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chunsheng Feng

Date

03/04/2016

Definition at line 430 of file pbcgs.c.

10.58.2.7 fasp_solver_dstr_pbcgs()

```
INT fasp_solver_dstr_pbcgs (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned BiCGstab method for solving Au=b.

Parameters

A	Pointer to the coefficient matrix
b	Pointer to the dvector of right hand side
u	Pointer to the dvector of DOFs
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou

Date

04/25/2010

Rewritten by Chensong Zhang on 04/30/2012 Modified by Feiteng Huang on 06/01/2012: fix restart param-init Modified by Chensong Zhang on 03/31/2013

Definition at line 2133 of file pbcgs.c.

10.58.2.8 fasp_solver_dstr_pvbcgs()

```
INT fasp_solver_dstr_pvbcgs (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned BiCGstab method for solving Au=b, Rewritten from Matlab 2011a.

Parameters

A	Pointer to the coefficient matrix
b	Pointer to the dvector of right hand side
u	Pointer to the dvector of DOFs
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chunsheng Feng

Date

03/04/2016

Definition at line 2473 of file pbcgs.c.

10.59 pbcgs_mf.c File Reference

Krylov subspace methods – Preconditioned BiCGstab (matrix free)

```
#include <math.h>
#include <float.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_pbcgs (mxv_matfree ∗mf, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned BiCGstab method for solving Au=b.
- INT fasp_solver_pvbcgs (mxv_matfree ∗mf, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned BiCGstab method for solving Au=b, Rewritten from Matlab 2011a.

10.59.1 Detailed Description

Krylov subspace methods – Preconditioned BiCGstab (matrix free)

Abstract algorithm of Krylov method

Krylov method to solve A∗x=b is to generate {x_k} to approximate x, where x_k is the optimal solution in Krylov space

V_k=span{r_0,A∗r_0,A^2∗r_0,...,A^{k-1}∗r_0},

under some inner product.

For the implementation, we generate a series of {p_k} such that V_k=span{p_1,...,p_k}. Details:

Step 0. Given A, b, x_0, M

Step 1. Compute residual r_0 = b-A∗x_0 and convergence check;

Step 2. Initialization z_0 = M^{-1}∗r_0, p_0=z_0;

Step 3. Main loop ...

FOR k = 0:MaxIt

- get step size alpha = f(r_k,z_k,p_k);
- update solution: x_{k+1} = x_k + alpha∗p_k;
- perform stagnation check;
- update residual: r_{k+1} = r_k - alpha∗(A∗p_k);
- perform residual check;
- obtain p_{k+1} using {p_0, p_1, ... , p_k};
- prepare for next iteration;
- print the result of k-th iteration; END FOR

Convergence check is: norm(r)/norm(b) < tol

Stagnation check is like following:

- IF norm(alpha∗p_k)/norm(x_{k+1}) < tol_stag
  1. compute r=b-A∗x_{k+1};
  2. convergence check;
  3. IF (not converged & restart_number < Max_Stag_Check) restart;
- END IF

Residual check is like following:

- IF norm(r_{k+1})/norm(b) < tol
  1. compute the real residual r = b-A∗x_{k+1};
  2. convergence check;
  3. IF (not converged & restart_number < Max_Res_Check) restart;
- END IF

Note

Refer to Y. Saad 2003 Iterative methods for sparse linear systems (2nd Edition), SIAM

10.59.2 Function Documentation

10.59.2.1 fasp_solver_pbcgs()

```
INT fasp_solver_pbcgs (
            mxv_matfree * mf,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned BiCGstab method for solving Au=b.

Parameters

mf	Pointer to mxv_matfree: the spmv operation
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

09/09/2009

Rewritten by Chensong Zhang on 04/30/2012 Modified by Feiteng Huang on 06/01/2012: fix restart param-init Modified by Feiteng Huang on 09/26/2012, (mmatrix free)

Definition at line 92 of file pbcgs_mf.c.

10.59.2.2 fasp_solver_pvbcgs()

```
INT fasp_solver_pvbcgs (
            mxv_matfree * mf,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned BiCGstab method for solving Au=b, Rewritten from Matlab 2011a.

Parameters

mf	Pointer to mxv_matfree: the spmv operation
b	Pointer to the dvector of right hand side
u	Pointer to the dvector of DOFs
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chunsheng Feng

Date

03/04/2016

Definition at line 431 of file pbcgs_mf.c.

10.60 pcg.c File Reference

Krylov subspace methods – Preconditioned conjugate gradient.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_dcsr_pcg (dCSRmat *A, dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned conjugate gradient method for solving Au=b.
- INT fasp_solver_dbsr_pcg (dBSRmat *A, dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned conjugate gradient method for solving Au=b.
- INT fasp_solver_dblc_pcg (dBLCmat *A, dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned conjugate gradient method for solving Au=b.
- INT fasp_solver_dstr_pcg (dSTRmat *A, dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned conjugate gradient method for solving Au=b.

10.60.1 Detailed Description

Krylov subspace methods – Preconditioned conjugate gradient.

Abstract algorithm

PCG method to solve A*x=b is to generate {x_k} to approximate x

Step 0. Given A, b, x_0, M

Step 1. Compute residual r_0 = b-A*x_0 and convergence check;

Step 2. Initialization z_0 = M^{-1}*r_0, p_0=z_0;

Step 3. Main loop ...

FOR k = 0:MaxIt

- get step size alpha = f(r_k,z_k,p_k);
- update solution: x_{k+1} = x_k + alpha∗p_k;
- perform stagnation check;
- update residual: r_{k+1} = r_k - alpha∗(A∗p_k);
- perform residual check;
- obtain p_{k+1} using {p_0, p_1, ... , p_k};
- prepare for next iteration;
- print the result of k-th iteration; END FOR

Convergence check: norm(r)/norm(b) < tol

Stagnation check:

- IF norm(alpha∗p_k)/norm(x_{k+1}) < tol_stag
  1. compute r=b-A∗x_{k+1};
  2. convergence check;
  3. IF (not converged & restart_number < Max_Stag_Check) restart;
- END IF

Residual check:

- IF norm(r_{k+1})/norm(b) < tol
  1. compute the real residual r = b-A∗x_{k+1};
  2. convergence check;
  3. IF (not converged & restart_number < Max_Res_Check) restart;
- END IF

Note

Refer to Y. Saad 2003 Iterative methods for sparse linear systems (2nd Edition), SIAM
See spcg.c for a safer version

10.60.2 Function Documentation

10.60.2.1 fasp_solver_dblc_pcg()

```
INT fasp_solver_dblc_pcg (
            dBLCmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned conjugate gradient method for solving Au=b.

Parameters

A	Pointer to dBLCmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

05/24/2010

Modified by Chensong Zhang on 04/30/2012 Modified by Chensong Zhang on 03/28/2013

Definition at line 665 of file pcg.c.

10.60.2.2 fasp_solver_dbsr_pcg()

```
INT fasp_solver_dbsr_pcg (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned conjugate gradient method for solving Au=b.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

05/26/2014

Definition at line 373 of file pcg.c.

10.60.2.3 fasp_solver_dcsr_pcg()

```
INT fasp_solver_dcsr_pcg (
            dCSRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned conjugate gradient method for solving Au=b.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang, Xiaozhe Hu, Shiquan Zhang

Date

05/06/2010

Modified by Chensong Zhang on 04/30/2012 Modified by Chensong Zhang on 03/28/2013

Definition at line 84 of file pcg.c.

10.60.2.4 fasp_solver_dstr_pcg()

```
INT fasp_solver_dstr_pcg (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned conjugate gradient method for solving Au=b.

Parameters

A	Pointer to dSTRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou

Date

04/25/2010

Modified by Chensong Zhang on 04/30/2012 Modified by Chensong Zhang on 03/28/2013

Definition at line 957 of file pcg.c.

10.61 pcg_mf.c File Reference

Krylov subspace methods – Preconditioned conjugate gradient (matrix free)

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_pcg (mxv_matfree ∗mf, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned conjugate gradient (CG) method for solving Au=b.

10.61.1 Detailed Description

Krylov subspace methods – Preconditioned conjugate gradient (matrix free)

Abstract algorithm

PCG method to solve A∗x=b is to generate {x_k} to approximate x

Step 0. Given A, b, x_0, M

Step 1. Compute residual r_0 = b-A∗x_0 and convergence check;

Step 2. Initialization z_0 = M^{-1}∗r_0, p_0=z_0;

Step 3. Main loop ...

FOR k = 0:MaxIt

- get step size alpha = f(r_k,z_k,p_k);
- update solution: x_{k+1} = x_k + alpha∗p_k;
- perform stagnation check;
- update residual: r_{k+1} = r_k - alpha∗(A∗p_k);
- perform residual check;
- obtain p_{k+1} using {p_0, p_1, ... , p_k};
- prepare for next iteration;
- print the result of k-th iteration; END FOR

Convergence check is: norm(r)/norm(b) < tol

Stagnation check is like following:

- IF norm(alpha*p_k)/norm(x_{k+1}) < tol_stag
    1. compute r=b-A*x_{k+1};
    2. convergence check;
    3. IF (not converged & restart_number < Max_Stag_Check) restart;
- END IF

Residual check is like following:

- IF norm(r_{k+1})/norm(b) < tol
    1. compute the real residual r = b-A*x_{k+1};
    2. convergence check;
    3. IF (not converged & restart_number < Max_Res_Check) restart;
- END IF

Note

Refer to Y. Saad 2003 Iterative methods for sparse linear systems (2nd Edition), SIAM

10.61.2 Function Documentation

10.61.2.1 fasp_solver_pcg()

```
INT fasp_solver_pcg (
            mxv_matfree * mf,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned conjugate gradient (CG) method for solving Au=b.

Parameters

mf	Pointer to mxv_matfree: the spmv operation
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang, Xiaozhe Hu, Shiquan Zhang

Date

05/06/2010

Modified by Chensong Zhang on 04/30/2012 Modified by Feiteng Huang on 09/19/2012: matrix free

Definition at line 86 of file pcg_mf.c.

10.62 pgcg.c File Reference

Krylov subspace methods – Preconditioned Generalized CG.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_dcsr_pgcg (dCSRmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned generilzed conjugate gradient (GCG) method for solving Au=b.

10.62.1 Detailed Description

Krylov subspace methods – Preconditioned Generalized CG.

Note

Refer to Concus, P. and Golub, G.H. and O'Leary, D.P. A Generalized Conjugate Gradient Method for the Numerical: Solution of Elliptic Partial Differential Equations, Computer Science Department, Stanford University, 1976

10.62.2 Function Documentation

10.62.2.1 fasp_solver_dcsr_pgcg()

```
INT fasp_solver_dcsr_pgcg (
            dCSRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned generilzed conjugate gradient (GCG) method for solving Au=b.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

01/01/2012

Modified by Chensong Zhang on 05/01/2012

Definition at line 44 of file pgcg.c.

10.63 pgcg_mf.c File Reference

Krylov subspace methods – Preconditioned Generalized CG (matrix free)

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_pgcg (mxv_matfree ∗mf, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned generilzed conjugate gradient (GCG) method for solving Au=b.

10.63.1 Detailed Description

Krylov subspace methods – Preconditioned Generalized CG (matrix free)

Note

Refer to Concus, P. and Golub, G.H. and O'Leary, D.P. A Generalized Conjugate Gradient Method for the Numerical: Solution of Elliptic Partial Differential Equations, Computer Science Department, Stanford University, 1976

10.63.2 Function Documentation

10.63.2.1 fasp_solver_pgcg()

```
INT fasp_solver_pgcg (
            mxv_matfree * mf,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned generilzed conjugate gradient (GCG) method for solving Au=b.

Parameters

mf	Pointer to mxv_matfree: the spmv operation
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type – Not implemented
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

01/01/2012

Note

Not completely implemented yet! –Chensong

Modified by Chensong Zhang on 05/01/2012 Modified by Feiteng Huang on 09/26/2012: matrix free

Definition at line 47 of file pgcg_mf.c.

10.64 pgcr.c File Reference

Krylov subspace methods – Preconditioned GCR.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_dcsr_pgcr (dCSRmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  A preconditioned GCR method for solving Au=b.
- INT fasp_solver_dcsr_pgcr1 (dCSRmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  A preconditioned GCR method for solving Au=b.

10.64.1 Detailed Description

Krylov subspace methods – Preconditioned GCR.

10.64.2 Function Documentation

10.64.2.1 fasp_solver_dcsr_pgcr()

```
INT fasp_solver_dcsr_pgcr (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

A preconditioned GCR method for solving Au=b.

Parameters

A	Pointer to the coefficient matrix
b	Pointer to the dvector of right hand side
x	Pointer to the dvector of dofs
pc	Pointer to the structure of precondition (precond)
tol	Tolerance for stopage
MaxIt	Maximal number of iterations
restart	Restart number for GCR
stop_type	Stopping type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Note

Refer to YVAN NOTAY "AN AGGREGATION-BASED ALGEBRAIC MULTIGRID METHOD"

Author

Zheng Li

Date

12/23/2014

Definition at line 37 of file pgcr.c.

10.64.2.2 fasp_solver_dcsr_pgcr1()

```
INT fasp_solver_dcsr_pgcr1 (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

A preconditioned GCR method for solving Au=b.

Parameters

A	Pointer to the coefficient matrix
b	Pointer to the dvector of right hand side
x	Pointer to the dvector of dofs
pc	Pointer to the structure of precondition (precond)
tol	Tolerance for stopage
MaxIt	Maximal number of iterations
restart	Restart number for GCR
stop_type	Stopping type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Lu Wang

Date

11/02/2014

Warning

Deprecated function. Remove it later!!! –Chensong

Definition at line 226 of file pgcr.c.

10.65 pgmres.c File Reference

Krylov subspace methods – Right-preconditioned GMRes.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_dcsr_pgmres (dCSRmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

    Right preconditioned GMRES method for solving Au=b.
- INT fasp_solver_dblc_pgmres (dBLCmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

    Preconditioned GMRES method for solving Au=b.
- INT fasp_solver_dbsr_pgmres (dBSRmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

    Preconditioned GMRES method for solving Au=b.
- INT fasp_solver_dstr_pgmres (dSTRmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

    Preconditioned GMRES method for solving Au=b.

10.65.1 Detailed Description

Krylov subspace methods – Right-preconditioned GMRes.

Note

Refer to Y. Saad 2003 Iterative methods for sparse linear systems (2nd Edition), SIAM
Four subroutines use the same algorithm for different matrix types!
See also pvgmres.c for a variable restarting version.
See spgmres.c for a safer version

10.65.2 Function Documentation

10.65.2.1 fasp_solver_dblc_pgmres()

```
INT fasp_solver_dblc_pgmres (
            dBLCmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned GMRES method for solving Au=b.

Parameters

A	Pointer to dBLCmat: the coefficient matrix
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns

Parameters

pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

05/24/2010

Modified by Chensong Zhang on 05/01/2012 Modified by Chensong Zhang on 04/05/2013: add stop_type and safe check

Definition at line 356 of file pgmres.c.

10.65.2.2 fasp_solver_dbsr_pgmres()

```
INT fasp_solver_dbsr_pgmres (
            dBSRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned GMRES method for solving Au=b.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou

Date

2010/12/21

Modified by Chensong Zhang on 05/01/2012 Modified by Chensong Zhang on 04/05/2013: add stop_type and safe check

Definition at line 659 of file pgmres.c.

10.65.2.3 fasp_solver_dcsr_pgmres()

```
INT fasp_solver_dcsr_pgmres (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Right preconditioned GMRES method for solving Au=b.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou

Date

2010/11/28

Modified by Chensong Zhang on 05/01/2012 Modified by Chensong Zhang on 04/05/2013: Add stop_type and safe check Modified by Chunsheng Feng on 07/22/2013: Add adapt memory allocate Modified by Chensong Zhang on 07/30/2014: Make memory allocation size long int Modified by Chensong Zhang on 09/21/2014: Add comments and reorganize code

Definition at line 53 of file pgmres.c.

10.65.2.4 fasp_solver_dstr_pgmres()

```
INT fasp_solver_dstr_pgmres (
            dSTRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned GMRES method for solving Au=b.

Parameters

A	Pointer to dSTRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou

Date

2010/11/28

Modified by Chensong Zhang on 05/01/2012 Modified by Chensong Zhang on 04/05/2013: add stop_type and safe check

Definition at line 963 of file pgmres.c.

10.66 pgmres_mf.c File Reference

Krylov subspace methods – Preconditioned GMRes (matrix free)

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_pgmres (mxv_matfree ∗mf, dvector ∗b, dvector ∗x, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Solve "Ax=b" using PGMRES (right preconditioned) iterative method.

10.66.1 Detailed Description

Krylov subspace methods – Preconditioned GMRes (matrix free)

Note

Refer to Y. Saad 2003 Iterative methods for sparse linear systems (2nd Edition), SIAM
Refer to A.H. Baker, E.R. Jessup, and Tz.V. Kolev A Simple Strategy for Varying the Restart Parameter in GMR↩ES(m) Journal of Computational and Applied Mathematics, 230 (2009) pp. 751-761. UCRL-JRNL-235266.

10.66.2 Function Documentation

10.66.2.1 fasp_solver_pgmres()

```
INT fasp_solver_pgmres (
            mxv_matfree * mf,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Solve "Ax=b" using PGMRES (right preconditioned) iterative method.

Parameters

mf	Pointer to mxv_matfree: the spmv operation
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type – DOES not support this parameter
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou

Date

2010/11/28

Modified by Chensong Zhang on 05/01/2012 Modified by Feiteng Huang on 09/26/2012: matrix free Modified by Chunsheng Feng on 07/22/2013: Add adapt memory allocate

Definition at line 50 of file pgmres_mf.c.

10.67 pminres.c File Reference

Krylov subspace methods – Preconditioned minimal residual.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_dcsr_pminres (dCSRmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  A preconditioned minimal residual (Minres) method for solving Au=b.
- INT fasp_solver_dblc_pminres (dBLCmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  A preconditioned minimal residual (Minres) method for solving Au=b.
- INT fasp_solver_dstr_pminres (dSTRmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  A preconditioned minimal residual (Minres) method for solving Au=b.

10.67.1 Detailed Description

Krylov subspace methods – Preconditioned minimal residual.

Abstract algorithm of Krylov method

Krylov method to solve A∗x=b is to generate {x_k} to approximate x, where x_k is the optimal solution in Krylov space

V_k=span{r_0,A∗r_0,A^2∗r_0,...,A^{k-1}∗r_0},

under some inner product.

For the implementation, we generate a series of {p_k} such that V_k=span{p_1,...,p_k}. Details:

Step 0. Given A, b, x_0, M

Step 1. Compute residual r_0 = b-A∗x_0 and convergence check;

Step 2. Initialization z_0 = M^{-1}∗r_0, p_0=z_0;

Step 3. Main loop ...

FOR k = 0:MaxIt

- get step size alpha = f(r_k,z_k,p_k);
- update solution: x_{k+1} = x_k + alpha∗p_k;
- perform stagnation check;
- update residual: r_{k+1} = r_k - alpha∗(A∗p_k);
- perform residual check;
- obtain p_{k+1} using {p_0, p_1, ... , p_k};
- prepare for next iteration;
- print the result of k-th iteration; END FOR

Convergence check: norm(r)/norm(b) < tol

Stagnation check:

- IF norm(alpha∗p_k)/norm(x_{k+1}) < tol_stag
  1. compute r=b-A∗x_{k+1};
  2. convergence check;
  3. IF (not converged & restart_number < Max_Stag_Check) restart;
- END IF

Residual check:

- IF norm(r_{k+1})/norm(b) < tol
  1. compute the real residual r = b-A∗x_{k+1};
  2. convergence check;
  3. IF (not converged & restart_number < Max_Res_Check) restart;
- END IF

Note

Refer to Y. Saad 2003 Iterative methods for sparse linear systems (2nd Edition), SIAM
See spminres.c for a safer version

10.67.2 Function Documentation

10.67.2.1 fasp_solver_dblc_pminres()

```
INT fasp_solver_dblc_pminres (
            dBLCmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

A preconditioned minimal residual (Minres) method for solving Au=b.

Parameters

A	Pointer to dBLCmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

05/01/2012

Note

Rewritten based on the original version by Xiaozhe Hu 05/24/2010

Modified by Chensong Zhang on 04/09/2013

Definition at line 499 of file pminres.c.

10.67.2.2 fasp_solver_dcsr_pminres()

```
INT fasp_solver_dcsr_pminres (
            dCSRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

A preconditioned minimal residual (Minres) method for solving Au=b.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

05/01/2012

Note

Rewritten based on the original version by Shiquan Zhang 05/10/2010

Modified by Chensong Zhang on 04/09/2013

Definition at line 92 of file pminres.c.

10.67.2.3 fasp_solver_dstr_pminres()

```
INT fasp_solver_dstr_pminres (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

A preconditioned minimal residual (Minres) method for solving Au=b.

Parameters

A	Pointer to dSTRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/09/2013

Definition at line 902 of file pminres.c.

10.68 pminres_mf.c File Reference

Krylov subspace methods – Preconditioned minimal residual (matrix free)

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_pminres (mxv_matfree *mf, dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  A preconditioned minimal residual (Minres) method for solving Au=b.

10.68.1 Detailed Description

Krylov subspace methods – Preconditioned minimal residual (matrix free)

Abstract algorithm of Krylov method

Krylov method to solve A*x=b is to generate {x_k} to approximate x, where x_k is the optimal solution in Krylov space

V_k=span{r_0,A*r_0,A^2*r_0,...,A^{k-1}*r_0},

under some inner product.

For the implementation, we generate a series of {p_k} such that V_k=span{p_1,...,p_k}. Details:

Step 0. Given A, b, x_0, M

Step 1. Compute residual r_0 = b-A*x_0 and convergence check;

Step 2. Initialization z_0 = M^{-1}*r_0, p_0=z_0;

Step 3. Main loop ...

FOR k = 0:MaxIt

- get step size alpha = f(r_k,z_k,p_k);
- update solution: x_{k+1} = x_k + alpha*p_k;
- perform stagnation check;
- update residual: r_{k+1} = r_k - alpha*(A*p_k);
- perform residual check;
- obtain p_{k+1} using {p_0, p_1, ... , p_k};
- prepare for next iteration;
- print the result of k-th iteration; END FOR

Convergence check is: norm(r)/norm(b) < tol

Stagnation check is like following:

- IF norm(alpha*p_k)/norm(x_{k+1}) < tol_stag
  1. compute r=b-A*x_{k+1};

2. convergence check;
3. IF (not converged & restart_number < Max_Stag_Check) restart;

- END IF

Residual check is like following:

- IF norm(r_{k+1})/norm(b) < tol
  1. compute the real residual r = b-A∗x_{k+1};
  2. convergence check;
  3. IF (not converged & restart_number < Max_Res_Check) restart;
- END IF

Note

Refer to Y. Saad 2003 Iterative methods for sparse linear systems (2nd Edition), SIAM

10.68.2 Function Documentation

10.68.2.1 fasp_solver_pminres()

```
INT fasp_solver_pminres (
            mxv_matfree * mf,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

A preconditioned minimal residual (Minres) method for solving Au=b.

Parameters

mf	Pointer to mxv_matfree: the spmv operation
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Shiquan Zhang

Date

10/24/2010

Rewritten by Chensong Zhang on 05/01/2012 Modified by Feiteng Huang on 09/26/2012: matrix free

Definition at line 89 of file pminres_mf.c.

10.69 precond_blc.c File Reference

Preconditioners for dBLCmat matrices.

```
#include "fasp.h"
#include "fasp_block.h"
#include "fasp_functs.h"
```

Functions

- void fasp_precond_block_diag_3 (REAL ∗r, REAL ∗z, void ∗data)
  block diagonal preconditioning (3x3 block matrix, each diagonal block is solved exactly)
- void fasp_precond_block_diag_3_amg (REAL ∗r, REAL ∗z, void ∗data)
  block diagonal preconditioning (3x3 block matrix, each diagonal block is solved by AMG)
- void fasp_precond_block_diag_4 (REAL ∗r, REAL ∗z, void ∗data)
  block diagonal preconditioning (4x4 block matrix, each diagonal block is solved exactly)
- void fasp_precond_block_lower_3 (REAL ∗r, REAL ∗z, void ∗data)
  block lower triangular preconditioning (3x3 block matrix, each diagonal block is solved exactly)
- void fasp_precond_block_lower_3_amg (REAL ∗r, REAL ∗z, void ∗data)
  block lower triangular preconditioning (3x3 block matrix, each diagonal block is solved by AMG)
- void fasp_precond_block_lower_4 (REAL ∗r, REAL ∗z, void ∗data)
  block lower triangular preconditioning (4x4 block matrix, each diagonal block is solved exactly)
- void fasp_precond_block_upper_3 (REAL ∗r, REAL ∗z, void ∗data)
  block upper triangular preconditioning (3x3 block matrix, each diagonal block is solved exactly)
- void fasp_precond_block_upper_3_amg (REAL ∗r, REAL ∗z, void ∗data)
  block upper triangular preconditioning (3x3 block matrix, each diagonal block is solved AMG)
- void fasp_precond_block_SGS_3 (REAL ∗r, REAL ∗z, void ∗data)
  block symmetric GS preconditioning (3x3 block matrix, each diagonal block is solved exactly)
- void fasp_precond_block_SGS_3_amg (REAL ∗r, REAL ∗z, void ∗data)
  block symmetric GS preconditioning (3x3 block matrix, each diagonal block is solved exactly)
- void fasp_precond_sweeping (REAL ∗r, REAL ∗z, void ∗data)
  sweeping preconditioner for Maxwell equations

10.69.1 Detailed Description

Preconditioners for dBLCmat matrices.

Note

Need to be cleaned up. –Chensong

10.69.2 Function Documentation

10.69.2.1 fasp_precond_block_diag_3()

```
void fasp_precond_block_diag_3 (
            REAL * r,
            REAL * z,
            void * data )
```

block diagonal preconditioning (3x3 block matrix, each diagonal block is solved exactly)

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

07/10/2014

Definition at line 28 of file precond_blc.c.

10.69.2.2 fasp_precond_block_diag_3_amg()

```
void fasp_precond_block_diag_3_amg (
            REAL * r,
            REAL * z,
            void * data )
```

block diagonal preconditioning (3x3 block matrix, each diagonal block is solved by AMG)

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

07/10/2014

Definition at line 112 of file precond_blc.c.

10.69.2.3 fasp_precond_block_diag_4()

```
void fasp_precond_block_diag_4 (
            REAL * r,
            REAL * z,
            void * data )
```

block diagonal preconditioning (4x4 block matrix, each diagonal block is solved exactly)

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

07/10/2014

Definition at line 177 of file precond_blc.c.

10.69.2.4 fasp_precond_block_lower_3()

```
void fasp_precond_block_lower_3 (
            REAL * r,
```

```
REAL * z,
void * data )
```

block lower triangular preconditioning (3x3 block matrix, each diagonal block is solved exactly)

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

07/10/2014

Definition at line 273 of file precond_blc.c.

10.69.2.5 fasp_precond_block_lower_3_amg()

```
void fasp_precond_block_lower_3_amg (
            REAL * r,
            REAL * z,
            void * data )
```

block lower triangular preconditioning (3x3 block matrix, each diagonal block is solved by AMG)

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

07/10/2014

Definition at line 355 of file precond_blc.c.

10.69.2.6 fasp_precond_block_lower_4()

```
void fasp_precond_block_lower_4 (
            REAL * r,
```

```
REAL * z,
void * data )
```

block lower triangular preconditioning (4x4 block matrix, each diagonal block is solved exactly)

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

07/10/2014

Definition at line 429 of file precond_blc.c.

10.69.2.7 fasp_precond_block_SGS_3()

```
void fasp_precond_block_SGS_3 (
            REAL * r,
            REAL * z,
            void * data )
```

block symmetric GS preconditioning (3x3 block matrix, each diagonal block is solved exactly)

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

02/19/2015

Definition at line 690 of file precond_blc.c.

10.69.2.8 fasp_precond_block_SGS_3_amg()

```
void fasp_precond_block_SGS_3_amg (
            REAL * r,
```

```
REAL * z,
void * data )
```

block symmetric GS preconditioning (3x3 block matrix, each diagonal block is solved exactly)

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

02/19/2015

Definition at line 806 of file precond_blc.c.

10.69.2.9 fasp_precond_block_upper_3()

```
void fasp_precond_block_upper_3 (
            REAL * r,
            REAL * z,
            void * data )
```

block upper triangular preconditioning (3x3 block matrix, each diagonal block is solved exactly)

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

02/18/2015

Definition at line 527 of file precond_blc.c.

10.69.2.10 fasp_precond_block_upper_3_amg()

```
void fasp_precond_block_upper_3_amg (
            REAL * r,
```

```
REAL * z,
void * data )
```

block upper triangular preconditioning (3x3 block matrix, each diagonal block is solved AMG)

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

02/19/2015

Definition at line 609 of file precond_blc.c.

10.69.2.11 fasp_precond_sweeping()

```
void fasp_precond_sweeping (
            REAL * r,
            REAL * z,
            void * data )
```

sweeping preconditioner for Maxwell equations

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

05/01/2014

Definition at line 916 of file precond_blc.c.

10.70 precond_bsr.c File Reference

Preconditioners for dBSRmat matrices.

```
#include "fasp.h"
#include "fasp_functs.h"
#include "mg_util.inl"
```

Functions

- void fasp_precond_dbsr_diag (REAL ∗r, REAL ∗z, void ∗data)
  Diagonal preconditioner z=inv(D)∗r.
- void fasp_precond_dbsr_diag_nc2 (REAL ∗r, REAL ∗z, void ∗data)
  Diagonal preconditioner z=inv(D)∗r.
- void fasp_precond_dbsr_diag_nc3 (REAL ∗r, REAL ∗z, void ∗data)
  Diagonal preconditioner z=inv(D)∗r.
- void fasp_precond_dbsr_diag_nc5 (REAL ∗r, REAL ∗z, void ∗data)
  Diagonal preconditioner z=inv(D)∗r.
- void fasp_precond_dbsr_diag_nc7 (REAL ∗r, REAL ∗z, void ∗data)
  Diagonal preconditioner z=inv(D)∗r.
- void fasp_precond_dbsr_ilu (REAL ∗r, REAL ∗z, void ∗data)
  ILU preconditioner.
- void fasp_precond_dbsr_ilu_mc_omp (REAL ∗r, REAL ∗z, void ∗data)
  Multi-threads Parallel ILU preconditioner based on graph coloring.
- void fasp_precond_dbsr_ilu_levsch_omp (REAL ∗r, REAL ∗z, void ∗data)
  Multi-threads Parallel ILU preconditioner based on level schedule strategy.
- void fasp_precond_dbsr_amg (REAL ∗r, REAL ∗z, void ∗data)
  AMG preconditioner.
- void fasp_precond_dbsr_nl_amli (REAL ∗r, REAL ∗z, void ∗data)
  Nonlinear AMLI-cycle AMG preconditioner.
- void fasp_precond_dbsr_amg_nk (REAL ∗r, REAL ∗z, void ∗data)
  AMG with extra near kernel solve preconditioner.

10.70.1 Detailed Description

Preconditioners for dBSRmat matrices.

10.70.2 Function Documentation

10.70.2.1 fasp_precond_dbsr_amg()

```
void fasp_precond_dbsr_amg (
            REAL * r,
            REAL * z,
            void * data )
```

AMG preconditioner.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

08/07/2011

Definition at line 970 of file precond_bsr.c.

10.70.2.2 fasp_precond_dbsr_amg_nk()

```
void fasp_precond_dbsr_amg_nk (
            REAL * r,
            REAL * z,
            void * data )
```

AMG with extra near kernel solve preconditioner.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

05/26/2014

Definition at line 1050 of file precond_bsr.c.

10.70.2.3 fasp_precond_dbsr_diag()

```
void fasp_precond_dbsr_diag (
            REAL * r,
```

```
REAL * z,
void * data )
```

Diagonal preconditioner z=inv(D)∗r.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Zhou Zhiyang, Xiaozhe Hu

Date

10/26/2010

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

05/24/2012

Note

Works for general nb (Xiaozhe)

Definition at line 37 of file precond_bsr.c.

10.70.2.4 fasp_precond_dbsr_diag_nc2()

```
void fasp_precond_dbsr_diag_nc2 (
            REAL * r,
            REAL * z,
            void * data )
```

Diagonal preconditioner z=inv(D)∗r.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Zhou Zhiyang, Xiaozhe Hu

Date

11/18/2011

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

05/24/2012

Note

Works for 2-component (Xiaozhe)

Definition at line 111 of file precond_bsr.c.

10.70.2.5 fasp_precond_dbsr_diag_nc3()

```
void fasp_precond_dbsr_diag_nc3 (
            REAL * r,
            REAL * z,
            void * data )
```

Diagonal preconditioner z=inv(D)∗r.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Zhou Zhiyang, Xiaozhe Hu

Date

01/06/2011

Modified by Chunsheng Feng Xiaoqiang Yue

Date

05/24/2012

Note

Works for 3-component (Xiaozhe)

Definition at line 161 of file precond_bsr.c.

10.70.2.6 fasp_precond_dbsr_diag_nc5()

```
void fasp_precond_dbsr_diag_nc5 (
            REAL * r,
            REAL * z,
            void * data )
```

Diagonal preconditioner z=inv(D)∗r.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Zhou Zhiyang, Xiaozhe Hu

Date

01/06/2011

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

05/24/2012

Note

Works for 5-component (Xiaozhe)

Definition at line 211 of file precond_bsr.c.

10.70.2.7 fasp_precond_dbsr_diag_nc7()

```
void fasp_precond_dbsr_diag_nc7 (
            REAL * r,
            REAL * z,
            void * data )
```

Diagonal preconditioner z=inv(D)∗r.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Zhou Zhiyang, Xiaozhe Hu

Date

01/06/2011

Modified by Chunsheng Feng Xiaoqiang Yue

Date

05/24/2012

Note

Works for 7-component (Xiaozhe)

Definition at line 260 of file precond_bsr.c.

10.70.2.8 fasp_precond_dbsr_ilu()

```
void fasp_precond_dbsr_ilu (
            REAL * r,
            REAL * z,
            void * data )
```

ILU preconditioner.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Shiquan Zhang, Xiaozhe Hu

Date

11/09/2010

Note

Works for general nb (Xiaozhe)

Definition at line 306 of file precond_bsr.c.

10.70.2.9 fasp_precond_dbsr_ilu_levsch_omp()

```
void fasp_precond_dbsr_ilu_levsch_omp (
            REAL * r,
            REAL * z,
            void * data )
```

Multi-threads Parallel ILU preconditioner based on level schedule strategy.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

ZhengLi

Date

12/04/2016

Note

Only works for nb 1, 2, and 3 (Zheng)

Definition at line 764 of file precond_bsr.c.

10.70.2.10 fasp_precond_dbsr_ilu_mc_omp()

```
void fasp_precond_dbsr_ilu_mc_omp (
            REAL * r,
            REAL * z,
            void * data )
```

Multi-threads Parallel ILU preconditioner based on graph coloring.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

ZhengLi

Date

12/04/2016

Note

Only works for nb 1, 2, and 3 (Zheng)

Definition at line 566 of file precond_bsr.c.

10.70.2.11 fasp_precond_dbsr_nl_amli()

```
void fasp_precond_dbsr_nl_amli (
            REAL * r,
            REAL * z,
            void * data )
```

Nonlinear AMLI-cycle AMG preconditioner.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

02/06/2012

Definition at line 1014 of file precond_bsr.c.

10.71 precond_csr.c File Reference

Preconditioners for dCSRmat matrices.

```
#include "fasp.h"
#include "fasp_functs.h"
#include "mg_util.inl"
```

Functions

- precond ∗ fasp_precond_setup (const SHORT precond_type, AMG_param ∗amgparam, ILU_param ∗iluparam, dCSRmat ∗A)

    Setup preconditioner interface for iterative methods.
- void fasp_precond_diag (REAL ∗r, REAL ∗z, void ∗data)

    Diagonal preconditioner z=inv(D)∗r.
- void fasp_precond_ilu (REAL ∗r, REAL ∗z, void ∗data)

    ILU preconditioner.
- void fasp_precond_ilu_forward (REAL ∗r, REAL ∗z, void ∗data)

    ILU preconditioner: only forward sweep.
- void fasp_precond_ilu_backward (REAL ∗r, REAL ∗z, void ∗data)

    ILU preconditioner: only backward sweep.
- void fasp_precond_Schwarz (REAL ∗r, REAL ∗z, void ∗data)

    get z from r by Schwarz
- void fasp_precond_amg (REAL ∗r, REAL ∗z, void ∗data)

    AMG preconditioner.
- void fasp_precond_famg (REAL ∗r, REAL ∗z, void ∗data)

    Full AMG preconditioner.
- void fasp_precond_amli (REAL ∗r, REAL ∗z, void ∗data)

    AMLI AMG preconditioner.
- void fasp_precond_nl_amli (REAL ∗r, REAL ∗z, void ∗data)

    Nonlinear AMLI AMG preconditioner.
- void fasp_precond_amg_nk (REAL ∗r, REAL ∗z, void ∗data)

    AMG with extra near kernel solve as preconditioner.
- void fasp_precond_free (const SHORT precond_type, precond ∗pc)

    free preconditioner

10.71.1 Detailed Description

Preconditioners for dCSRmat matrices.

10.71.2 Function Documentation

10.71.2.1 fasp_precond_amg()

```
void fasp_precond_amg (
            REAL * r,
            REAL * z,
            void * data )
```

AMG preconditioner.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Chensong Zhang

Date

04/06/2010

Definition at line 397 of file precond_csr.c.

10.71.2.2 fasp_precond_amg_nk()

```
void fasp_precond_amg_nk (
            REAL * r,
            REAL * z,
            void * data )
```

AMG with extra near kernel solve as preconditioner.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

05/26/2014

Definition at line 532 of file precond_csr.c.

10.71.2.3 fasp_precond_amli()

```
void fasp_precond_amli (
            REAL * r,
```

```
REAL * z,
void * data )
```

AMLI AMG preconditioner.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

01/23/2011

Definition at line 466 of file precond_csr.c.

10.71.2.4 fasp_precond_diag()

```
void fasp_precond_diag (
            REAL * r,
            REAL * z,
            void * data )
```

Diagonal preconditioner z=inv(D)∗r.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Chensong Zhang

Date

04/06/2010

Definition at line 156 of file precond_csr.c.

10.71.2.5 fasp_precond_famg()

```
void fasp_precond_famg (
            REAL * r,
```

```
REAL * z,
void * data )
```

Full AMG preconditioner.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

02/27/2011

Definition at line 433 of file precond_csr.c.

10.71.2.6 fasp_precond_free()

```
void fasp_precond_free (
            const SHORT precond_type,
            precond * pc )
```

free preconditioner

Parameters

precond_type	Preconditioner type
∗*pc*	precondition data & fct

Returns

void

Author

Feiteng Huang

Date

12/24/2012

Definition at line 616 of file precond_csr.c.

10.71.2.7 fasp_precond_ilu()

```
void fasp_precond_ilu (
            REAL * r,
            REAL * z,
            void * data )
```

ILU preconditioner.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Shiquan Zhang

Date

04/06/2010

Definition at line 182 of file precond_csr.c.

10.71.2.8 fasp_precond_ilu_backward()

```
void fasp_precond_ilu_backward (
            REAL * r,
            REAL * z,
            void * data )
```

ILU preconditioner: only backward sweep.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu, Shiquan Zhang

Date

04/06/2010

Definition at line 299 of file precond_csr.c.

10.71.2.9 fasp_precond_ilu_forward()

```
void fasp_precond_ilu_forward (
            REAL * r,
            REAL * z,
            void * data )
```

ILU preconditioner: only forward sweep.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu, Shiquang Zhang

Date

04/06/2010

Definition at line 246 of file precond_csr.c.

10.71.2.10 fasp_precond_nl_amli()

```
void fasp_precond_nl_amli (
            REAL * r,
            REAL * z,
            void * data )
```

Nonlinear AMLI AMG preconditioner.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

04/25/2011

Definition at line 499 of file precond_csr.c.

10.71.2.11 fasp_precond_Schwarz()

```
void fasp_precond_Schwarz (
            REAL * r,
            REAL * z,
            void * data )
```

get z from r by Schwarz

Parameters

*r	pointer to residual
*z	pointer to preconditioned residual
*data	pointer to precondition data

Author

Xiaozhe Hu

Date

03/22/2010

Note

Change Schwarz interface by Zheng Li on 11/18/2014

Definition at line 352 of file precond_csr.c.

10.71.2.12 fasp_precond_setup()

```
precond * fasp_precond_setup (
            const SHORT precond_type,
            AMG_param * amgparam,
            ILU_param * iluparam,
            dCSRmat * A )
```

Setup preconditioner interface for iterative methods.

Parameters

precond_type	Preconditioner type
amgparam	Pointer to AMG parameters
iluparam	Pointer to ILU parameters
A	Pointer to the coefficient matrix

Returns

Pointer to preconditioner

Author

Feiteng Huang

Date

05/18/2009

Definition at line 30 of file precond_csr.c.

10.72 precond_str.c File Reference

Preconditioners for dSTRmat matrices.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_precond_dstr_diag (REAL ∗r, REAL ∗z, void ∗data)

  Diagonal preconditioner z=inv(D)∗r.
- void fasp_precond_dstr_ilu0 (REAL ∗r, REAL ∗z, void ∗data)

  Preconditioning using STR_ILU(0) decomposition.
- void fasp_precond_dstr_ilu1 (REAL ∗r, REAL ∗z, void ∗data)

  Preconditioning using STR_ILU(1) decomposition.
- void fasp_precond_dstr_ilu0_forward (REAL ∗r, REAL ∗z, void ∗data)

  Preconditioning using STR_ILU(0) decomposition: Lz = r.
- void fasp_precond_dstr_ilu0_backward (REAL ∗r, REAL ∗z, void ∗data)

  Preconditioning using STR_ILU(0) decomposition: Uz = r.
- void fasp_precond_dstr_ilu1_forward (REAL ∗r, REAL ∗z, void ∗data)

  Preconditioning using STR_ILU(1) decomposition: Lz = r.
- void fasp_precond_dstr_ilu1_backward (REAL ∗r, REAL ∗z, void ∗data)

  Preconditioning using STR_ILU(1) decomposition: Uz = r.
- void fasp_precond_dstr_blockgs (REAL ∗r, REAL ∗z, void ∗data)

  CPR-type preconditioner (STR format)

10.72.1 Detailed Description

Preconditioners for dSTRmat matrices.

10.72.2 Function Documentation

10.72.2.1 fasp_precond_dstr_blockgs()

```
void fasp_precond_dstr_blockgs (
            REAL * r,
            REAL * z,
            void * data )
```

CPR-type preconditioner (STR format)

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Shiquan Zhang

Date

10/17/2010

Definition at line 1706 of file precond_str.c.

10.72.2.2 fasp_precond_dstr_diag()

```
void fasp_precond_dstr_diag (
            REAL * r,
            REAL * z,
            void * data )
```

Diagonal preconditioner z=inv(D)∗r.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Shiquan Zhang

Date

04/06/2010

Definition at line 27 of file precond_str.c.

10.72.2.3 fasp_precond_dstr_ilu0()

```
void fasp_precond_dstr_ilu0 (
            REAL * r,
            REAL * z,
            void * data )
```

Preconditioning using STR_ILU(0) decomposition.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Shiquan Zhang

Date

04/21/2010

Definition at line 54 of file precond_str.c.

10.72.2.4 fasp_precond_dstr_ilu0_backward()

```
void fasp_precond_dstr_ilu0_backward (
            REAL * r,
            REAL * z,
            void * data )
```

Preconditioning using STR_ILU(0) decomposition: Uz = r.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Shiquan Zhang

Date

06/07/2010

Definition at line 978 of file precond_str.c.

10.72.2.5 fasp_precond_dstr_ilu0_forward()

```
void fasp_precond_dstr_ilu0_forward (
            REAL * r,
            REAL * z,
            void * data )
```

Preconditioning using STR_ILU(0) decomposition: Lz = r.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Shiquan Zhang

Date

06/07/2010

Definition at line 815 of file precond_str.c.

10.72.2.6 fasp_precond_dstr_ilu1()

```
void fasp_precond_dstr_ilu1 (
            REAL * r,
            REAL * z,
            void * data )
```

Preconditioning using STR_ILU(1) decomposition.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Shiquan Zhang

Date

04/21/2010

Definition at line 336 of file precond_str.c.

10.72.2.7 fasp_precond_dstr_ilu1_backward()

```
void fasp_precond_dstr_ilu1_backward (
            REAL * r,
            REAL * z,
            void * data )
```

Preconditioning using STR_ILU(1) decomposition: Uz = r.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Shiquan Zhang

Date

04/21/2010

Definition at line 1425 of file precond_str.c.

10.72.2.8 fasp_precond_dstr_ilu1_forward()

```
void fasp_precond_dstr_ilu1_forward (
            REAL * r,
            REAL * z,
            void * data )
```

Preconditioning using STR_ILU(1) decomposition: Lz = r.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Shiquan Zhang

Date

04/21/2010

Definition at line 1159 of file precond_str.c.

10.73 pvfgmres.c File Reference

Krylov subspace methods – Preconditioned variable-restarting flexible GMRes.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_dcsr_pvfgmres (dCSRmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Solve "Ax=b" using PFGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during the iteration and flexible preconditioner can be used.
- INT fasp_solver_dbsr_pvfgmres (dBSRmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Solve "Ax=b" using PFGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during the iteration and flexible preconditioner can be used.
- INT fasp_solver_dblc_pvfgmres (dBLCmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Solve "Ax=b" using PFGMRES (right preconditioned) iterative method in which the restart parameter can be adaptively modified during the iteration and flexible preconditioner can be used.

10.73.1 Detailed Description

Krylov subspace methods – Preconditioned variable-restarting flexible GMRes.

Note

Refer to Y. Saad 2003 Iterative methods for sparse linear systems (2nd Edition), SIAM
Refer to A.H. Baker, E.R. Jessup, and Tz.V. Kolev A Simple Strategy for Varying the Restart Parameter in GMR↩ES(m) Journal of Computational and Applied Mathematics, 230 (2009) pp. 751-761. UCRL-JRNL-235266.
This file is modifed from pvgmres.c

10.73.2 Function Documentation

10.73.2.1 fasp_solver_dblc_pvfgmres()

```
INT fasp_solver_dblc_pvfgmres (
            dBLCmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Solve "Ax=b" using PFGMRES (right preconditioned) iterative method in which the restart parameter can be adaptively modified during the iteration and flexible preconditioner can be used.

Parameters

∗A	pointer to the coefficient matrix
∗b	pointer to the right hand side vector
∗x	pointer to the solution vector
MaxIt	maximal iteration number allowed
tol	tolerance
∗pc	pointer to preconditioner data
prtlvl	How much information to print out
stop_type	default stopping criterion,i.e.||r_k||/||r_0||<tol, is used.
restart	number of restart for GMRES

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

01/04/2012

Note

Based on Zhiyang Zhou's pvgmres.c

Modified by Chunsheng Feng on 07/22/2013: Add adaptive memory allocate Modified by Chensong Zhang on 05/09/2015: Clean up for stopping types

Definition at line 712 of file pvfgmres.c.

10.73.2.2 fasp_solver_dbsr_pvfgmres()

```
INT fasp_solver_dbsr_pvfgmres (
            dBSRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Solve "Ax=b" using PFGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during the iteration and flexible preconditioner can be used.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type – DOES not support this parameter
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

02/05/2012

Modified by Chensong Zhang on 05/01/2012 Modified by Chunsheng Feng on 07/22/2013: Add adaptive memory allocate Modified by Chensong Zhang on 05/09/2015: Clean up for stopping types

Definition at line 382 of file pvfgmres.c.

10.73.2.3 fasp_solver_dcsr_pvfgmres()

```
INT fasp_solver_dcsr_pvfgmres (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Solve "Ax=b" using PFGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during the iteration and flexible preconditioner can be used.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type – DOES not support this parameter
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

01/04/2012

Modified by Chensong Zhang on 05/01/2012 Modified by Chunsheng Feng on 07/22/2013: Add adaptive memory allocate Modified by Chensong Zhang on 05/09/2015: Clean up for stopping types

Definition at line 54 of file pvfgmres.c.

10.74 pvfgmres_mf.c File Reference

Krylov subspace methods – Preconditioned variable-restarting flexible GMRes (matrix free)

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_pvfgmres (mxv_matfree *mf, dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

    Solve "Ax=b" using PFGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during the iteration and flexible preconditioner can be used.

10.74.1 Detailed Description

Krylov subspace methods – Preconditioned variable-restarting flexible GMRes (matrix free)

Note

Refer to Y. Saad 2003 Iterative methods for sparse linear systems (2nd Edition), SIAM
Refer to A.H. Baker, E.R. Jessup, and Tz.V. Kolev A Simple Strategy for Varying the Restart Parameter in GMRES(m) Journal of Computational and Applied Mathematics, 230 (2009) pp. 751-761. UCRL-JRNL-235266.
This file is modifed from pvgmres.c

10.74.2 Function Documentation

10.74.2.1 fasp_solver_pvfgmres()

```
INT fasp_solver_pvfgmres (
            mxv_matfree * mf,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Solve "Ax=b" using PFGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during the iteration and flexible preconditioner can be used.

Parameters

mf	Pointer to mxv_matfree: the spmv operation
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type – DOES not support this parameter
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

01/04/2012

Modified by Chensong Zhang on 05/01/2012 Modified by Feiteng Huang on 09/26/2012: matrix free Modified by Chunsheng Feng on 07/22/2013: Add adapt memory allocate

Definition at line 55 of file pvfgmres_mf.c.

10.75 pvgmres.c File Reference

Krylov subspace methods – Preconditioned variable-restart GMRes.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_dcsr_pvgmres (dCSRmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Right preconditioned GMRES method in which the restart parameter can be adaptively modified during the iteration.
- INT fasp_solver_dblc_pvgmres (dBLCmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Right preconditioned GMRES method in which the restart parameter can be adaptively modified during the iteration.
- INT fasp_solver_dbsr_pvgmres (dBSRmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Right preconditioned GMRES method in which the restart parameter can be adaptively modified during the iteration.
- INT fasp_solver_dstr_pvgmres (dSTRmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Right preconditioned GMRES method in which the restart parameter can be adaptively modified during the iteration.

10.75.1 Detailed Description

Krylov subspace methods – Preconditioned variable-restart GMRes.

Note

Refer to A.H. Baker, E.R. Jessup, and Tz.V. Kolev A Simple Strategy for Varying the Restart Parameter in GMR↩ES(m) Journal of Computational and Applied Mathematics, 230 (2009) pp. 751-761. UCRL-JRNL-235266.
See spvgmres.c for a safer version

10.75.2 Function Documentation

10.75.2.1 fasp_solver_dblc_pvgmres()

```
INT fasp_solver_dblc_pvgmres (
            dBLCmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Right preconditioned GMRES method in which the restart parameter can be adaptively modified during the iteration.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/05/2013

Definition at line 393 of file pvgmres.c.

10.75.2.2 fasp_solver_dbsr_pvgmres()

```
INT fasp_solver_dbsr_pvgmres (
            dBSRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Right preconditioned GMRES method in which the restart parameter can be adaptively modified during the iteration.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou

Date

12/21/2011

Modified by Chensong Zhang on 05/01/2012 Modified by Chensong Zhang on 04/06/2013: Add stop type support

Definition at line 738 of file pvgmres.c.

10.75.2.3 fasp_solver_dcsr_pvgmres()

```
INT fasp_solver_dcsr_pvgmres (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Right preconditioned GMRES method in which the restart parameter can be adaptively modified during the iteration.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou

Date

2010/12/14

Modified by Chensong Zhang on 12/13/2011 Modified by Chensong Zhang on 05/01/2012 Modified by Chensong Zhang on 04/06/2013: Add stop type support Modified by Chunsheng Feng on 07/22/2013: Add adapt memory allocate

Definition at line 51 of file pvgmres.c.

10.75.2.4 fasp_solver_dstr_pvgmres()

```
INT fasp_solver_dstr_pvgmres (
            dSTRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Right preconditioned GMRES method in which the restart parameter can be adaptively modified during the iteration.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou

Date

2010/12/14

Modified by Chensong Zhang on 05/01/2012 Modified by Chensong Zhang on 04/06/2013: Add stop type support

Definition at line 1083 of file pvgmres.c.

10.76 pvgmres_mf.c File Reference

Krylov subspace methods – Preconditioned variable-restarting GMRes (matrix free)

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_pvgmres (mxv_matfree *mf, dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Solve "Ax=b" using PGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during the iteration.

10.76.1 Detailed Description

Krylov subspace methods – Preconditioned variable-restarting GMRes (matrix free)

Note

Refer to A.H. Baker, E.R. Jessup, and Tz.V. Kolev A Simple Strategy for Varying the Restart Parameter in GMR↩ES(m) Journal of Computational and Applied Mathematics, 230 (2009) pp. 751-761. UCRL-JRNL-235266.

10.76.2 Function Documentation

10.76.2.1 fasp_solver_pvgmres()

```
INT fasp_solver_pvgmres (
            mxv_matfree * mf,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Solve "Ax=b" using PGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during the iteration.

Parameters

mf	Pointer to mxv_matfree: the spmv operation
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns
pc	Pointer to precond: the structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type – DOES not support this parameter
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou

Date

2010/12/14

Modified by Chensong Zhang on 12/13/2011 Modified by Chensong Zhang on 05/01/2012 Modified by Feiteng Huang on 09/26/2012: matrix free Modified by Chunsheng Feng on 07/22/2013: Add adapt memory allocate

Definition at line 54 of file pvgmres_mf.c.

10.77 quadrature.c File Reference

Quadrature rules.

```
#include <stdio.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_quad2d (const INT num_qp, const INT ncoor, REAL(*quad)[3])

    Initialize Lagrange quadrature points and weights.
- void fasp_gauss2d (const INT num_qp, const INT ncoor, REAL(*gauss)[3])

    Initialize Gauss quadrature points and weights.

10.77.1 Detailed Description

Quadrature rules.

10.77.2 Function Documentation

10.77.2.1 fasp_gauss2d()

```
void fasp_gauss2d (
            const INT num_qp,
            const INT ncoor,
            REAL(*) gauss[3] )
```

Initialize Gauss quadrature points and weights.

Parameters

num_qp	Number of quadrature points
ncoor	Dimension of space
gauss	Quadrature points and weight

Author

Xuehai Huang, Chensong Zhang, Ludmil Zikatanov

Date

10/21/2008

Note

gauss[∗][0] – quad point x in ref coor gauss[∗][1] – quad point y in ref coor gauss[∗][2] – quad weight

Definition at line 210 of file quadrature.c.

10.77.2.2 fasp_quad2d()

```
void fasp_quad2d (
            const INT num_qp,
            const INT ncoor,
            REAL(*) quad[3] )
```

Initialize Lagrange quadrature points and weights.

Parameters

num_qp	Number of quadrature points
ncoor	Dimension of space
quad	Quadrature points and weights

Author

Xuehai Huang, Chensong Zhang, Ludmil Zikatanov

Date

10/21/2008

Note

quad[∗][0] – quad point x in ref coor quad[∗][1] – quad point y in ref coor quad[∗][2] – quad weight

Definition at line 31 of file quadrature.c.

10.78 rap.c File Reference

Tripple-matrix multiplication R∗A∗P.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- dCSRmat fasp_blas_dcsr_rap2 (INT ∗ir, INT ∗jr, REAL ∗r, INT ∗ia, INT ∗ja, REAL ∗a, INT ∗ipt, INT ∗jpt, REAL ∗pt, INT n, INT nc, INT ∗maxrpout, INT ∗ipin, INT ∗jpin)

  Compute R∗A∗P.

10.78.1 Detailed Description

Tripple-matrix multiplication R∗A∗P.

C-version by Ludmil Zikatanov 2010-04-08

tested 2010-04-08

10.78.2 Function Documentation

10.78.2.1 fasp_blas_dcsr_rap2()

```
dCSRmat fasp_blas_dcsr_rap2 (
            INT * ir,
            INT * jr,
            REAL * r,
            INT * ia,
            INT * ja,
            REAL * a,
            INT * ipt,
            INT * jpt,
            REAL * pt,
            INT n,
            INT nc,
            INT * maxrpout,
            INT * ipin,
            INT * jpin )
```

Compute R∗A∗P.

Author

Ludmil Zikatanov

Date

04/08/2010

Note

It uses dCSRmat only. The functions called from here are in sparse_util.c

Definition at line 33 of file rap.c.

10.79 schwarz_setup.c File Reference

Setup phase for the Schwarz methods.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "mg_util.inl"
```

Functions

- void fasp_Schwarz_get_block_matrix (Schwarz_data ∗Schwarz, INT nblk, INT ∗iblock, INT ∗jblock, INT ∗mask)

  Form Schwarz partition data.
- INT fasp_Schwarz_setup (Schwarz_data ∗Schwarz, Schwarz_param ∗param)

  Setup phase for the Schwarz methods.
- void fasp_dcsr_Schwarz_forward_smoother (Schwarz_data ∗Schwarz, Schwarz_param ∗param, dvector ∗x, dvector ∗b)

  Schwarz smoother: forward sweep.
- void fasp_dcsr_Schwarz_backward_smoother (Schwarz_data ∗Schwarz, Schwarz_param ∗param, dvector ∗x, dvector ∗b)

  Schwarz smoother: backward sweep.

10.79.1 Detailed Description

Setup phase for the Schwarz methods.

10.79.2 Function Documentation

10.79.2.1 fasp_dcsr_Schwarz_backward_smoother()

```
void fasp_dcsr_Schwarz_backward_smoother (
            Schwarz_data * Schwarz,
            Schwarz_param * param,
            dvector * x,
            dvector * b )
```

Schwarz smoother: backward sweep.

Parameters

Schwarz	Pointer to the Schwarz data
param	Pointer to the Schwarz parameter
x	Pointer to solution vector
b	Pointer to right hand

Author

Zheng Li, Chensong Zhang

Date

2014/10/5

Definition at line 404 of file schwarz_setup.c.

10.79.2.2 fasp_dcsr_Schwarz_forward_smoother()

```
void fasp_dcsr_Schwarz_forward_smoother (
            Schwarz_data * Schwarz,
            Schwarz_param * param,
            dvector * x,
            dvector * b )
```

Schwarz smoother: forward sweep.

Parameters

Schwarz	Pointer to the Schwarz data
param	Pointer to the Schwarz parameter
x	Pointer to solution vector
b	Pointer to right hand

Author

Zheng Li, Chensong Zhang

Date

2014/10/5

Definition at line 294 of file schwarz_setup.c.

10.79.2.3 fasp_Schwarz_get_block_matrix()

```
void fasp_Schwarz_get_block_matrix (
            Schwarz_data * Schwarz,
            INT nblk,
            INT * iblock,
            INT * jblock,
            INT * mask )
```

Form Schwarz partition data.

Parameters

Schwarz	Pointer to the Schwarz data
nblk	Number of partitions
iblock	Pointer to number of vertices on each level
jblock	Pointer to vertices of each level
mask	Pointer to flag array

Author

Zheng Li, Chensong Zhang

Date

2014/09/29

Definition at line 34 of file schwarz_setup.c.

10.79.2.4 fasp_Schwarz_setup()

```
INT fasp_Schwarz_setup (
            Schwarz_data * Schwarz,
            Schwarz_param * param )
```

Setup phase for the Schwarz methods.

Parameters

Schwarz	Pointer to the Schwarz data
param	Type of the Schwarz method

Returns

FASP_SUCCESS if succeed

Author

Ludmil, Xiaozhe Hu

Date

03/22/2011

Modified by Zheng Li on 10/09/2014

Definition at line 125 of file schwarz_setup.c.

10.80 smat.c File Reference

Simple operations for *small* dense matrices in row-major format.

```
#include "fasp.h"
#include "fasp_functs.h"
```

Macros

- #define SWAP(a, b) {temp=(a);(a)=(b);(b)=temp;}

Functions

- void fasp_blas_smat_inv_nc2 (REAL ∗a)
  Compute the inverse matrix of a 2∗2 full matrix A (in place)
- void fasp_blas_smat_inv_nc3 (REAL ∗a)
  Compute the inverse matrix of a 3∗3 full matrix A (in place)
- void fasp_blas_smat_inv_nc4 (REAL ∗a)
  Compute the inverse matrix of a 4∗4 full matrix A (in place)
- void fasp_blas_smat_inv_nc5 (REAL ∗a)
  Compute the inverse matrix of a 5∗5 full matrix A (in place)
- void fasp_blas_smat_inv_nc7 (REAL ∗a)
  Compute the inverse matrix of a 7∗7 matrix a.
- void fasp_blas_smat_inv_nc (REAL ∗a, const INT n)
  Compute the inverse of a matrix using Gauss Elimination.
- void fasp_blas_smat_invp_nc (REAL ∗a, const INT n)
  Compute the inverse of a matrix using Gauss Elimination with Pivoting.
- INT fasp_blas_smat_inv (REAL ∗a, const INT n)
  Compute the inverse matrix of a small full matrix a.
- REAL fasp_blas_smat_Linfinity (REAL ∗A, const INT n)
  Compute the L infinity norm of A.
- void fasp_iden_free (idenmat ∗A)

Free idenmat sparse matrix data memeory space.

- void fasp_smat_identity_nc2 (REAL ∗a)

    Set a 2∗2 full matrix to be a identity.
- void fasp_smat_identity_nc3 (REAL ∗a)

    Set a 3∗3 full matrix to be a identity.
- void fasp_smat_identity_nc5 (REAL ∗a)

    Set a 5∗5 full matrix to be a identity.
- void fasp_smat_identity_nc7 (REAL ∗a)

    Set a 7∗7 full matrix to be a identity.
- void fasp_smat_identity (REAL ∗a, const INT n, const INT n2)

    Set a n∗n full matrix to be a identity.

10.80.1 Detailed Description

Simple operations for *small* dense matrices in row-major format.

10.80.2 Macro Definition Documentation

10.80.2.1 SWAP

```
#define SWAP(
              a,
              b ) {temp=(a);(a)=(b);(b)=temp;}
```

swap two numbers

Definition at line 9 of file smat.c.

10.80.3 Function Documentation

10.80.3.1 fasp_blas_smat_inv()

```
INT fasp_blas_smat_inv (
             REAL * a,
             const INT n )
```

Compute the inverse matrix of a small full matrix a.

Parameters

a	Pointer to the REAL array which stands a n∗n matrix
n	Dimension of the matrix

Author

Xiaozhe Hu, Shiquan Zhang

Date

04/21/2010

Definition at line 554 of file smat.c.

10.80.3.2 fasp_blas_smat_inv_nc()

```
void fasp_blas_smat_inv_nc (
            REAL * a,
            const INT n )
```

Compute the inverse of a matrix using Gauss Elimination.

Parameters

a	Pointer to the REAL array which stands a n∗n matrix
n	Dimension of the matrix

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 405 of file smat.c.

10.80.3.3 fasp_blas_smat_inv_nc2()

```
void fasp_blas_smat_inv_nc2 (
            REAL * a )
```

Compute the inverse matrix of a 2∗2 full matrix A (in place)

Parameters

a	Pointer to the REAL array which stands a 2∗2 matrix

Author

Xiaozhe Hu

Date

18/11/2011

Definition at line 25 of file smat.c.

10.80.3.4 fasp_blas_smat_inv_nc3()

```
void fasp_blas_smat_inv_nc3 (
            REAL * a )
```

Compute the inverse matrix of a 3∗3 full matrix A (in place)

Parameters

a	Pointer to the REAL array which stands a 3∗3 matrix

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 61 of file smat.c.

10.80.3.5 fasp_blas_smat_inv_nc4()

```
void fasp_blas_smat_inv_nc4 (
            REAL * a )
```

Compute the inverse matrix of a 4∗4 full matrix A (in place)

Parameters

a	Pointer to the REAL array which stands a 4∗4 matrix

Author

Xiaozhe Hu

Date

01/12/2013

Modified by Hongxuan Zhang on 06/13/2014: Fix a bug in M23.

Definition at line 115 of file smat.c.

10.80.3.6 fasp_blas_smat_inv_nc5()

```
void fasp_blas_smat_inv_nc5 (
            REAL * a )
```

Compute the inverse matrix of a 5∗5 full matrix A (in place)

Parameters

a	Pointer to the REAL array which stands a 5∗5 matrix

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 173 of file smat.c.

10.80.3.7 fasp_blas_smat_inv_nc7()

```
void fasp_blas_smat_inv_nc7 (
            REAL * a )
```

Compute the inverse matrix of a 7∗7 matrix a.

Parameters

a	Pointer to the REAL array which stands a 7∗7 matrix

Note

This is NOT implemented yet!

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 389 of file smat.c.

10.80.3.8 fasp_blas_smat_invp_nc()

```
void fasp_blas_smat_invp_nc (
            REAL * a,
            const INT n )
```

Compute the inverse of a matrix using Gauss Elimination with Pivoting.

Parameters

a	Pointer to the REAL array which stands a n*n matrix
n	Dimension of the matrix

Author

Chensong Zhang

Date

04/03/2015

Note

This routine is based on gaussj() from "Numerical Recipies in C"!

Definition at line 472 of file smat.c.

10.80.3.9 fasp_blas_smat_Linfinity()

```
REAL fasp_blas_smat_Linfinity (
            REAL * A,
            const INT n )
```

Compute the L infinity norm of A.

Parameters

A	Pointer to the n∗n dense matrix
n	the dimension of the dense matrix

Author

Xiaozhe Hu

Date

05/26/2014

Definition at line 595 of file smat.c.

10.80.3.10 fasp_iden_free()

```
void fasp_iden_free (
            idenmat * A )
```

Free idenmat sparse matrix data memeory space.

Parameters

A	Pointer to the idenmat matrix

Author

Chensong Zhang

Date

2010/04/03

Definition at line 628 of file smat.c.

10.80.3.11 fasp_smat_identity()

```
void fasp_smat_identity (
            REAL * a,
            const INT n,
            const INT n2 )
```

Set a n∗n full matrix to be a identity.

Parameters

a	Pointer to the REAL vector which stands for a n∗n full matrix
n	Size of full matrix
n2	Length of the REAL vector which stores the n∗n full matrix

Author

Xiaozhe Hu

Date

2010/12/25

Definition at line 728 of file smat.c.

10.80.3.12 fasp_smat_identity_nc2()

```
void fasp_smat_identity_nc2 (
            REAL * a )
```

Set a 2∗2 full matrix to be a identity.

Parameters

a	Pointer to the REAL vector which stands for a 2∗2 full matrix

Author

Xiaozhe Hu

Date

2011/11/18

Definition at line 648 of file smat.c.

10.80.3.13 fasp_smat_identity_nc3()

```
void fasp_smat_identity_nc3 (
            REAL * a )
```

Set a 3∗3 full matrix to be a identity.

Parameters

a	Pointer to the REAL vector which stands for a 3∗3 full matrix

Author

Xiaozhe Hu

Date

2010/12/25

Definition at line 665 of file smat.c.

10.80.3.14 fasp_smat_identity_nc5()

```
void fasp_smat_identity_nc5 (
            REAL * a )
```

Set a 5∗5 full matrix to be a identity.

Parameters

a	Pointer to the REAL vector which stands for a 5∗5 full matrix

Author

Xiaozhe Hu

Date

2010/12/25

Definition at line 682 of file smat.c.

10.80.3.15 fasp_smat_identity_nc7()

```
void fasp_smat_identity_nc7 (
            REAL * a )
```

Set a 7∗7 full matrix to be a identity.

Parameters

a	Pointer to the REAL vector which stands for a 7∗7 full matrix

Author

Xiaozhe Hu

Date

2010/12/25

Definition at line 703 of file smat.c.

10.81 smoother_bsr.c File Reference

Smoothers for dBSRmat matrices.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_smoother_dbsr_jacobi (dBSRmat *A, dvector *b, dvector *u)

  Jacobi relaxation.
- void fasp_smoother_dbsr_jacobi_setup (dBSRmat *A, dvector *b, dvector *u, REAL *diaginv)

  Setup for jacobi relaxation, fetch the diagonal sub-block matrixes and make them inverse first.
- void fasp_smoother_dbsr_jacobi1 (dBSRmat *A, dvector *b, dvector *u, REAL *diaginv)

  Jacobi relaxation.
- void fasp_smoother_dbsr_gs (dBSRmat *A, dvector *b, dvector *u, INT order, INT *mark)

  Gauss-Seidel relaxation.
- void fasp_smoother_dbsr_gs1 (dBSRmat *A, dvector *b, dvector *u, INT order, INT *mark, REAL *diaginv)

  Gauss-Seidel relaxation.
- void fasp_smoother_dbsr_gs_ascend (dBSRmat *A, dvector *b, dvector *u, REAL *diaginv)

  Gauss-Seidel relaxation in the ascending order.
- void fasp_smoother_dbsr_gs_ascend1 (dBSRmat *A, dvector *b, dvector *u)

  Gauss-Seidel relaxation in the ascending order.
- void fasp_smoother_dbsr_gs_descend (dBSRmat *A, dvector *b, dvector *u, REAL *diaginv)

  Gauss-Seidel relaxation in the descending order.
- void fasp_smoother_dbsr_gs_descend1 (dBSRmat *A, dvector *b, dvector *u)

  Gauss-Seidel relaxation in the descending order.
- void fasp_smoother_dbsr_gs_order1 (dBSRmat *A, dvector *b, dvector *u, REAL *diaginv, INT *mark)

  Gauss-Seidel relaxation in the user-defined order.
- void fasp_smoother_dbsr_gs_order2 (dBSRmat *A, dvector *b, dvector *u, INT *mark, REAL *work)

  Gauss-Seidel relaxation in the user-defined order.
- void fasp_smoother_dbsr_sor (dBSRmat *A, dvector *b, dvector *u, INT order, INT *mark, REAL weight)

  SOR relaxation.

- void fasp_smoother_dbsr_sor1 (dBSRmat ∗A, dvector ∗b, dvector ∗u, INT order, INT ∗mark, REAL ∗diaginv, REAL weight)

  SOR relaxation.
- void fasp_smoother_dbsr_sor_ascend (dBSRmat ∗A, dvector ∗b, dvector ∗u, REAL ∗diaginv, REAL weight)

  SOR relaxation in the ascending order.
- void fasp_smoother_dbsr_sor_descend (dBSRmat ∗A, dvector ∗b, dvector ∗u, REAL ∗diaginv, REAL weight)

  SOR relaxation in the descending order.
- void fasp_smoother_dbsr_sor_order (dBSRmat ∗A, dvector ∗b, dvector ∗u, REAL ∗diaginv, INT ∗mark, REAL weight)

  SOR relaxation in the user-defined order.
- void fasp_smoother_dbsr_ilu (dBSRmat ∗A, dvector ∗b, dvector ∗x, void ∗data)

  ILU method as the smoother in solving Au=b with multigrid method.

Variables

- REAL ilu_solve_omp = 0.0

10.81.1 Detailed Description

Smoothers for dBSRmat matrices.

10.81.2 Function Documentation

10.81.2.1 fasp_smoother_dbsr_gs()

```
void fasp_smoother_dbsr_gs (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            INT order,
            INT * mark )
```

Gauss-Seidel relaxation.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
order	Flag to indicate the order for smoothing If mark = NULL ASCEND 12: in ascending order DESCEND 21: in descending order If mark != NULL: in the user-defined order
mark	Pointer to NULL or to the user-defined ordering

Author

Zhiyang Zhou

Date

2010/10/25

Modified by Chunsheng Feng, Zheng Li on 08/03/2012

Definition at line 413 of file smoother_bsr.c.

10.81.2.2 fasp_smoother_dbsr_gs1()

```
void fasp_smoother_dbsr_gs1 (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            INT order,
            INT * mark,
            REAL * diaginv )
```

Gauss-Seidel relaxation.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
order	Flag to indicate the order for smoothing If mark = NULL ASCEND 12: in ascending order DESCEND 21: in descending order If mark != NULL: in the user-defined order
mark	Pointer to NULL or to the user-defined ordering
diaginv	Inverses for all the diagonal blocks of A

Author

Zhiyang Zhou

Date

2010/10/25

Definition at line 533 of file smoother_bsr.c.

10.81.2.3 fasp_smoother_dbsr_gs_ascend()

```
void fasp_smoother_dbsr_gs_ascend (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv )
```

Gauss-Seidel relaxation in the ascending order.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial guess, OUT: approximation)
diaginv	Inverses for all the diagonal blocks of A

Author

Zhiyang Zhou

Date

2010/10/25

Definition at line 570 of file smoother_bsr.c.

10.81.2.4 fasp_smoother_dbsr_gs_ascend1()

```
void fasp_smoother_dbsr_gs_ascend1 (
            dBSRmat * A,
            dvector * b,
            dvector * u )
```

Gauss-Seidel relaxation in the ascending order.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial guess, OUT: approximation)

Author

Xiaozhe

Date

01/01/2014

Note

The only difference between the functions 'fasp_smoother_dbsr_gs_ascend1' and 'fasp_smoother_dbsr_gs_↩ ascend' is that we don't have to multiply by the inverses of the diagonal blocks in each ROW since matrix A has been such scaled that all the diagonal blocks become identity matrices.

Definition at line 643 of file smoother_bsr.c.

10.81.2.5 fasp_smoother_dbsr_gs_descend()

```
void fasp_smoother_dbsr_gs_descend (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv )
```

Gauss-Seidel relaxation in the descending order.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial guess, OUT: approximation)
diaginv	Inverses for all the diagonal blocks of A

Author

Zhiyang Zhou

Date

2010/10/25

Definition at line 714 of file smoother_bsr.c.

10.81.2.6 fasp_smoother_dbsr_gs_descend1()

```
void fasp_smoother_dbsr_gs_descend1 (
            dBSRmat * A,
            dvector * b,
            dvector * u )
```

Gauss-Seidel relaxation in the descending order.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial guess, OUT: approximation)

Author

Xiaozhe Hu

Date

01/01/2014

Note

The only difference between the functions 'fasp_smoother_dbsr_gs_ascend1' and 'fasp_smoother_dbsr_gs_↩ ascend' is that we don't have to multiply by the inverses of the diagonal blocks in each ROW since matrix A has been such scaled that all the diagonal blocks become identity matrices.

Definition at line 788 of file smoother_bsr.c.

10.81.2.7 fasp_smoother_dbsr_gs_order1()

```
void fasp_smoother_dbsr_gs_order1 (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv,
            INT * mark )
```

Gauss-Seidel relaxation in the user-defined order.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial guess, OUT: approximation)
diaginv	Inverses for all the diagonal blocks of A
mark	Pointer to the user-defined ordering

Author

Zhiyang Zhou

Date

2010/10/25

Definition at line 860 of file smoother_bsr.c.

10.81.2.8 fasp_smoother_dbsr_gs_order2()

```
void fasp_smoother_dbsr_gs_order2 (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            INT * mark,
            REAL * work )
```

Gauss-Seidel relaxation in the user-defined order.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial guess, OUT: approximation)
mark	Pointer to the user-defined ordering
work	Work temp array

Author

Zhiyang Zhou

Date

2010/11/08

Note

The only difference between the functions 'fasp_smoother_dbsr_gs_order2' and 'fasp_smoother_dbsr_gs_order1' lies in that we don't have to multiply by the inverses of the diagonal blocks in each ROW since matrix A has been such scaled that all the diagonal blocks become identity matrices.

Definition at line 938 of file smoother_bsr.c.

10.81.2.9 fasp_smoother_dbsr_ilu()

```
void fasp_smoother_dbsr_ilu (
            dBSRmat * A,
            dvector * b,
            dvector * x,
            void * data )
```

ILU method as the smoother in solving Au=b with multigrid method.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
data	Pointer to user defined data

Author

Zhiyang Zhou, Zheng Li

Date

2010/10/25

NOTE: Add multi-threads parallel ILU block by Zheng Li 12/04/2016. form residual zr = b - A x

solve LU z=zr

x=x+z

Definition at line 1569 of file smoother_bsr.c.

10.81.2.10 fasp_smoother_dbsr_jacobi()

```
void fasp_smoother_dbsr_jacobi (
            dBSRmat * A,
            dvector * b,
            dvector * u )
```

Jacobi relaxation.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)

Author

Zhiyang Zhou

Date

2010/10/25

Modified by Chunsheng Feng, Zheng Li on 08/02/2012

Definition at line 35 of file smoother_bsr.c.

10.81.2.11 fasp_smoother_dbsr_jacobi1()

```
void fasp_smoother_dbsr_jacobi1 (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv )
```

Jacobi relaxation.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
diaginv	Inverses for all the diagonal blocks of A

Author

Zhiyang Zhou

Date

2010/10/25

Modified by Chunsheng Feng, Zheng Li on 08/03/2012

Definition at line 259 of file smoother_bsr.c.

10.81.2.12 fasp_smoother_dbsr_jacobi_setup()

```
void fasp_smoother_dbsr_jacobi_setup (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv )
```

Setup for jacobi relaxation, fetch the diagonal sub-block matrixes and make them inverse first.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
diaginv	Inverse of the diagonal entries

Author

Zhiyang Zhou

Date

10/25/2010

Modified by Chunsheng Feng, Zheng Li on 08/02/2012

Definition at line 150 of file smoother_bsr.c.

10.81.2.13 fasp_smoother_dbsr_sor()

```
void fasp_smoother_dbsr_sor (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            INT order,
            INT * mark,
            REAL weight )
```

SOR relaxation.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial guess, OUT: approximation)
order	Flag to indicate the order for smoothing If mark = NULL ASCEND 12: in ascending order DESCEND 21: in descending order If mark != NULL: in the user-defined order
mark	Pointer to NULL or to the user-defined ordering
weight	Over-relaxation weight

Author

Zhiyang Zhou

Date

2010/10/25

Modified by Chunsheng Feng, Zheng Li on 08/03/2012

Definition at line 1015 of file smoother_bsr.c.

10.81.2.14 fasp_smoother_dbsr_sor1()

```
void fasp_smoother_dbsr_sor1 (
        dBSRmat * A,
        dvector * b,
        dvector * u,
        INT order,
        INT * mark,
        REAL * diaginv,
        REAL weight )
```

SOR relaxation.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial guess, OUT: approximation)
order	Flag to indicate the order for smoothing If mark = NULL ASCEND 12: in ascending order DESCEND 21: in descending order If mark != NULL: in the user-defined order
mark	Pointer to NULL or to the user-defined ordering
diaginv	Inverses for all the diagonal blocks of A
weight	Over-relaxation weight

Author

Zhiyang Zhou

Date

2010/10/25

Definition at line 1137 of file smoother_bsr.c.

10.81.2.15 fasp_smoother_dbsr_sor_ascend()

```
void fasp_smoother_dbsr_sor_ascend (
        dBSRmat * A,
        dvector * b,
        dvector * u,
        REAL * diaginv,
        REAL weight )
```

SOR relaxation in the ascending order.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial guess, OUT: approximation)
diaginv	Inverses for all the diagonal blocks of A
weight	Over-relaxation weight

Author

Zhiyang Zhou

Date

2010/10/25

Modified by Chunsheng Feng, Zheng Li on 2012/09/04

Definition at line 1178 of file smoother_bsr.c.

10.81.2.16 fasp_smoother_dbsr_sor_descend()

```
void fasp_smoother_dbsr_sor_descend (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv,
            REAL weight )
```

SOR relaxation in the descending order.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial guess, OUT: approximation)
diaginv	Inverses for all the diagonal blocks of A
weight	Over-relaxation weight

Author

Zhiyang Zhou

Date

2010/10/25

Modified by Chunsheng Feng, Zheng Li on 2012/09/04

Definition at line 1307 of file smoother_bsr.c.

10.81.2.17 fasp_smoother_dbsr_sor_order()

```
void fasp_smoother_dbsr_sor_order (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv,
            INT * mark,
            REAL weight )
```

SOR relaxation in the user-defined order.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
diaginv	Inverses for all the diagonal blocks of A
mark	Pointer to the user-defined ordering
weight	Over-relaxation weight

Author

Zhiyang Zhou

Date

2010/10/25

Modified by Chunsheng Feng, Zheng Li on 2012/09/04

Definition at line 1438 of file smoother_bsr.c.

10.81.3 Variable Documentation

10.81.3.1 ilu_solve_omp

```
REAL ilu_solve_omp = 0.0
```

ILU time for the SOLVE phase

Definition at line 15 of file smoother_bsr.c.

10.82 smoother_csr.c File Reference

Smoothers for dCSRmat matrices.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_smoother_dcsr_jacobi (dvector ∗u, const INT i_1, const INT i_n, const INT s, dCSRmat ∗A, dvector ∗b, INT L)

    Jacobi method as a smoother.
- void fasp_smoother_dcsr_gs (dvector ∗u, const INT i_1, const INT i_n, const INT s, dCSRmat ∗A, dvector ∗b, INT L)

    Gauss-Seidel method as a smoother.
- void fasp_smoother_dcsr_gs_cf (dvector ∗u, dCSRmat ∗A, dvector ∗b, INT L, INT ∗mark, const INT order)

    Gauss-Seidel smoother with C/F ordering for Au=b.
- void fasp_smoother_dcsr_sgs (dvector ∗u, dCSRmat ∗A, dvector ∗b, INT L)

    Symmetric Gauss-Seidel method as a smoother.
- void fasp_smoother_dcsr_sor (dvector ∗u, const INT i_1, const INT i_n, const INT s, dCSRmat ∗A, dvector ∗b, INT L, const REAL w)

    SOR method as a smoother.
- void fasp_smoother_dcsr_sor_cf (dvector ∗u, dCSRmat ∗A, dvector ∗b, INT L, const REAL w, INT ∗mark, const INT order)

    SOR smoother with C/F ordering for Au=b.
- void fasp_smoother_dcsr_ilu (dCSRmat ∗A, dvector ∗b, dvector ∗x, void ∗data)

    ILU method as a smoother.
- void fasp_smoother_dcsr_kaczmarz (dvector ∗u, const INT i_1, const INT i_n, const INT s, dCSRmat ∗A, dvector ∗b, INT L, const REAL w)

    Kaczmarz method as a smoother.
- void fasp_smoother_dcsr_L1diag (dvector ∗u, const INT i_1, const INT i_n, const INT s, dCSRmat ∗A, dvector ∗b, INT L)

    Diagonal scaling (using L1 norm) as a smoother.
- void fasp_smoother_dcsr_gs_rb3d (dvector ∗u, dCSRmat ∗A, dvector ∗b, INT L, const INT order, INT ∗mark, const INT maximap, const INT nx, const INT ny, const INT nz)

    Colored Gauss-Seidel smoother for Au=b.

10.82.1 Detailed Description

Smoothers for dCSRmat matrices.

10.82.2 Function Documentation

10.82.2.1 fasp_smoother_dcsr_gs()

```
void fasp_smoother_dcsr_gs (
            dvector * u,
            const INT i_1,
            const INT i_n,
            const INT s,
            dCSRmat * A,
            dvector * b,
            INT L )
```

Gauss-Seidel method as a smoother.

Parameters

u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
i_1	Starting index
i_n	Ending index
s	Increasing step
A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
L	Number of iterations

Author

Xuehai Huang, Chensong Zhang

Date

09/26/2009

Modified by Chunsheng Feng, Zheng Li on 09/01/2012

Definition at line 195 of file smoother_csr.c.

10.82.2.2 fasp_smoother_dcsr_gs_cf()

```
void fasp_smoother_dcsr_gs_cf (
            dvector * u,
            dCSRmat * A,
            dvector * b,
            INT L,
            INT * mark,
            const INT order )
```

Gauss-Seidel smoother with C/F ordering for Au=b.

Parameters

u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
L	Number of iterations
mark	C/F marker array
order	C/F ordering: -1: F-first; 1: C-first

Author

Zhiyang Zhou

Date

11/12/2010

Modified by Chunsheng Feng, Xiaoqiang Yue on 05/24/2012

Definition at line 364 of file smoother_csr.c.

10.82.2.3 fasp_smoother_dcsr_gs_rb3d()

```
void fasp_smoother_dcsr_gs_rb3d (
            dvector * u,
            dCSRmat * A,
            dvector * b,
            INT L,
            const INT order,
            INT * mark,
            const INT maximap,
            const INT nx,
            const INT ny,
            const INT nz )
```

Colored Gauss-Seidel smoother for Au=b.

Parameters

u	Initial guess and the new approximation to the solution
A	Pointer to stiffness matrix
b	Pointer to right hand side
L	Number of iterations
order	Ordering: -1: Forward; 1: Backward
mark	Marker for C/F points
maximap	Size of IMAP
nx	Number vertex of X direction
ny	Number vertex of Y direction
nz	Number vertex of Z direction

Author

Chunsheng Feng

Date

02/08/2012

Definition at line 1425 of file smoother_csr.c.

10.82.2.4 fasp_smoother_dcsr_ilu()

```
void fasp_smoother_dcsr_ilu (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            void * data )
```

ILU method as a smoother.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
data	Pointer to user defined data

Author

Shiquan Zhang, Xiaozhe Hu

Date

2010/11/12

form residual zr = b - A x

Definition at line 1067 of file smoother_csr.c.

10.82.2.5 fasp_smoother_dcsr_jacobi()

```
void fasp_smoother_dcsr_jacobi (
            dvector * u,
            const INT i_1,
            const INT i_n,
            const INT s,
            dCSRmat * A,
            dvector * b,
            INT L )
```

Jacobi method as a smoother.

Parameters

u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
i_1	Starting index
i_n	Ending index
s	Increasing step
A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
L	Number of iterations

Author

Xuehai Huang, Chensong Zhang

Date

09/26/2009

Modified by Chunsheng Feng, Zheng Li on 08/29/2012

Definition at line 59 of file smoother_csr.c.

10.82.2.6 fasp_smoother_dcsr_kaczmarz()

```
void fasp_smoother_dcsr_kaczmarz (
            dvector * u,
            const INT i_1,
            const INT i_n,
            const INT s,
            dCSRmat * A,
            dvector * b,
            INT L,
            const REAL w )
```

Kaczmarz method as a smoother.

Parameters

u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
i_1	Starting index
i_n	Ending index
s	Increasing step
A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
L	Number of iterations
w	Over-relaxation weight

Author

Xiaozhe Hu

Date

2010/11/12

Modified by Chunsheng Feng, Zheng Li on 2012/09/01

Definition at line 1145 of file smoother_csr.c.

10.82.2.7 fasp_smoother_dcsr_L1diag()

```
void fasp_smoother_dcsr_L1diag (
            dvector * u,
            const INT i_1,
            const INT i_n,
            const INT s,
            dCSRmat * A,
            dvector * b,
            INT L )
```

Diagonal scaling (using L1 norm) as a smoother.

Parameters

u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
i_1	Starting index
i_n	Ending index
s	Increasing step
A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
L	Number of iterations

Author

Xiaozhe Hu, James Brannick

Date

01/26/2011

Modified by Chunsheng Feng, Zheng Li on 09/01/2012

Definition at line 1286 of file smoother_csr.c.

10.82.2.8 fasp_smoother_dcsr_sgs()

```
void fasp_smoother_dcsr_sgs (
            dvector * u,
            dCSRmat * A,
            dvector * b,
            INT L )
```

Symmetric Gauss-Seidel method as a smoother.

Parameters

u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
L	Number of iterations

Author

Xiaozhe Hu

Date

10/26/2010

Modified by Chunsheng Feng, Zheng Li on 09/01/2012

Definition at line 629 of file smoother_csr.c.

10.82.2.9 fasp_smoother_dcsr_sor()

```
void fasp_smoother_dcsr_sor (
            dvector * u,
            const INT i_1,
            const INT i_n,
            const INT s,
            dCSRmat * A,
            dvector * b,
            INT L,
            const REAL w )
```

SOR method as a smoother.

Parameters

u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
i_1	Starting index

Parameters

i_n	Ending index
s	Increasing step
A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
L	Number of iterations
w	Over-relaxation weight

Author

Xiaozhe Hu

Date

10/26/2010

Modified by Chunsheng Feng, Zheng Li on 09/01/2012

Definition at line 745 of file smoother_csr.c.

10.82.2.10 fasp_smoother_dcsr_sor_cf()

```
void fasp_smoother_dcsr_sor_cf (
            dvector * u,
            dCSRmat * A,
            dvector * b,
            INT L,
            const REAL w,
            INT * mark,
            const INT order )
```

SOR smoother with C/F ordering for Au=b.

Parameters

u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
L	Number of iterations
w	Over-relaxation weight
mark	C/F marker array
order	C/F ordering: -1: F-first; 1: C-first

Author

Zhiyang Zhou

Date

2010/11/12

Modified by Chunsheng Feng, Zheng Li on 08/29/2012

Definition at line 873 of file smoother_csr.c.

10.83 smoother_csr_cr.c File Reference

Smoothers for dCSRmat matrices using compatible relaxation.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_smoother_dcsr_gscr (INT pt, INT n, REAL ∗u, INT ∗ia, INT ∗ja, REAL ∗a, REAL ∗b, INT L, INT ∗CF)
  Gauss Seidel method restriced to a block.

10.83.1 Detailed Description

Smoothers for dCSRmat matrices using compatible relaxation.

Note

Restricted-smoothers for compatible relaxation, C/F smoothing, etc.

10.83.2 Function Documentation

10.83.2.1 fasp_smoother_dcsr_gscr()

```
void fasp_smoother_dcsr_gscr (
            INT pt,
            INT n,
            REAL * u,
            INT * ia,
            INT * ja,
            REAL * a,
            REAL * b,
            INT L,
            INT * CF )
```

Gauss Seidel method restriced to a block.

Parameters

pt	Relax type, e.g., cpt, fpt, etc..
n	Number of variables
u	Iterated solution
ia	Row pointer
ja	Column index
a	Pointers to sparse matrix values in CSR format
b	Pointer to right hand side – remove later also as MG relaxation on error eqn
L	Number of iterations
CF	Marker for C, F points

Author

James Brannick

Date

09/07/2010

Note

Gauss Seidel CR smoother (Smoother_Type = 99)

Definition at line 38 of file smoother_csr_cr.c.

10.84 smoother_csr_poly.c File Reference

Smoothers for dCSRmat matrices using poly. approx. to A^{-1}.

```
#include <math.h>
#include <time.h>
#include <float.h>
#include <limits.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_smoother_dcsr_poly (dCSRmat ∗Amat, dvector ∗brhs, dvector ∗usol, INT n, INT ndeg, INT L)

  poly approx to A^{-1} as MG smoother
- void fasp_smoother_dcsr_poly_old (dCSRmat ∗Amat, dvector ∗brhs, dvector ∗usol, INT n, INT ndeg, INT L)

  poly approx to A^{-1} as MG smoother: JK&LTZ2010

10.84.1 Detailed Description

Smoothers for dCSRmat matrices using poly. approx. to A^{-1}.

Refer to Johannes K. Kraus, Panayot S. Vassilevski, Ludmil T. Zikatanov "Polynomial of best uniform approximation to x^{-1} and smoothing in two-leve methods", 2013.

10.84.2 Function Documentation

10.84.2.1 fasp_smoother_dcsr_poly()

```
void fasp_smoother_dcsr_poly (
            dCSRmat * Amat,
            dvector * brhs,
            dvector * usol,
            INT n,
            INT ndeg,
            INT L )
```

poly approx to A^{-1} as MG smoother

Parameters

Amat	Pointer to stiffness matrix, consider square matrix.
brhs	Pointer to right hand side
usol	Pointer to solution
n	Problem size
ndeg	Degree of poly
L	Number of iterations

Author

Fei Cao, Xiaozhe Hu

Date

05/24/2012

Definition at line 48 of file smoother_csr_poly.c.

10.84.2.2 fasp_smoother_dcsr_poly_old()

```
void fasp_smoother_dcsr_poly_old (
            dCSRmat * Amat,
            dvector * brhs,
```

```
dvector * usol,
INT n,
INT ndeg,
INT L )
```

poly approx to A^{-1} as MG smoother: JK&LTZ2010

Parameters

Amat	Pointer to stiffness matrix
brhs	Pointer to right hand side
usol	Pointer to solution
n	Problem size
ndeg	Degree of poly
L	Number of iterations

Author

James Brannick and Ludmil T Zikatanov

Date

06/28/2010

Modified by Chunsheng Feng, Zheng Li on 10/18/2012

Definition at line 148 of file smoother_csr_poly.c.

10.85 smoother_str.c File Reference

Smoothers for dSTRmat matrices.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_smoother_dstr_jacobi (dSTRmat ∗A, dvector ∗b, dvector ∗u)

  Jacobi method as the smoother.
- void fasp_smoother_dstr_jacobi1 (dSTRmat ∗A, dvector ∗b, dvector ∗u, REAL ∗diaginv)

  Jacobi method as the smoother with diag_inv given.
- void fasp_smoother_dstr_gs (dSTRmat ∗A, dvector ∗b, dvector ∗u, const INT order, INT ∗mark)

  Gauss-Seidel method as the smoother.
- void fasp_smoother_dstr_gs1 (dSTRmat ∗A, dvector ∗b, dvector ∗u, const INT order, INT ∗mark, REAL ∗diaginv)

  Gauss-Seidel method as the smoother with diag_inv given.
- void fasp_smoother_dstr_gs_ascend (dSTRmat ∗A, dvector ∗b, dvector ∗u, REAL ∗diaginv)

  Gauss-Seidel method as the smoother in the ascending manner.
- void fasp_smoother_dstr_gs_descend (dSTRmat ∗A, dvector ∗b, dvector ∗u, REAL ∗diaginv)

  Gauss-Seidel method as the smoother in the descending manner.
- void fasp_smoother_dstr_gs_order (dSTRmat ∗A, dvector ∗b, dvector ∗u, REAL ∗diaginv, INT ∗mark)

Gauss method as the smoother in the user-defined order.

- void fasp_smoother_dstr_gs_cf (dSTRmat ∗A, dvector ∗b, dvector ∗u, REAL ∗diaginv, INT ∗mark, const INT order)

    Gauss method as the smoother in the C-F manner.
- void fasp_smoother_dstr_sor (dSTRmat ∗A, dvector ∗b, dvector ∗u, const INT order, INT ∗mark, const REAL weight)

    SOR method as the smoother.
- void fasp_smoother_dstr_sor1 (dSTRmat ∗A, dvector ∗b, dvector ∗u, const INT order, INT ∗mark, REAL ∗diaginv, const REAL weight)

    SOR method as the smoother.
- void fasp_smoother_dstr_sor_ascend (dSTRmat ∗A, dvector ∗b, dvector ∗u, REAL ∗diaginv, REAL weight)

    SOR method as the smoother in the ascending manner.
- void fasp_smoother_dstr_sor_descend (dSTRmat ∗A, dvector ∗b, dvector ∗u, REAL ∗diaginv, REAL weight)

    SOR method as the smoother in the descending manner.
- void fasp_smoother_dstr_sor_order (dSTRmat ∗A, dvector ∗b, dvector ∗u, REAL ∗diaginv, INT ∗mark, REAL weight)

    SOR method as the smoother in the user-defined order.
- void fasp_smoother_dstr_sor_cf (dSTRmat ∗A, dvector ∗b, dvector ∗u, REAL ∗diaginv, INT ∗mark, const INT order, const REAL weight)

    SOR method as the smoother in the C-F manner.
- void fasp_generate_diaginv_block (dSTRmat ∗A, ivector ∗neigh, dvector ∗diaginv, ivector ∗pivot)

    Generate inverse of diagonal block for block smoothers.
- void fasp_smoother_dstr_schwarz (dSTRmat ∗A, dvector ∗b, dvector ∗u, dvector ∗diaginv, ivector ∗pivot, ivector ∗neigh, ivector ∗order)

    Schwarz method as the smoother.

10.85.1 Detailed Description

Smoothers for dSTRmat matrices.

10.85.2 Function Documentation

10.85.2.1 fasp_generate_diaginv_block()

```
void fasp_generate_diaginv_block (
            dSTRmat * A,
            ivector * neigh,
            dvector * diaginv,
            ivector * pivot )
```

Generate inverse of diagonal block for block smoothers.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
neigh	Pointer to ivector: neighborhoods
diaginv	Pointer to dvector: the inverse of the diagonals
pivot	Pointer to ivector: the pivot of diagonal blocks

Author

Xiaozhe Hu

Date

10/01/2011

Definition at line 1521 of file smoother_str.c.

10.85.2.2 fasp_smoother_dstr_gs()

```
void fasp_smoother_dstr_gs (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            const INT order,
            INT * mark )
```

Gauss-Seidel method as the smoother.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
order	Flag to indicate the order for smoothing If mark = NULL ASCEND 12: in ascending manner DESCEND 21: in descending manner If mark != NULL USERDEFINED 0 : in the user-defined manner CPFIRST 1 : C-points first and then F-points FPFIRST -1 : F-points first and then C-points
mark	Pointer to the user-defined ordering(when order=0) or CF_marker array(when order!=0)

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 203 of file smoother_str.c.

10.85.2.3 fasp_smoother_dstr_gs1()

```
void fasp_smoother_dstr_gs1 (
            dSTRmat * A,
            dvector * b,
```

```
dvector * u,
const INT order,
INT * mark,
REAL * diaginv )
```

Gauss-Seidel method as the smoother with diag_inv given.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
order	Flag to indicate the order for smoothing If mark = NULL ASCEND 12: in ascending manner DESCEND 21: in descending manner If mark != NULL USERDEFINED 0 : in the user-defined manner CPFIRST 1 : C-points first and then F-points FPFIRST -1 : F-points first and then C-points
mark	Pointer to the user-defined ordering(when order=0) or CF_marker array(when order!=0)
diaginv	All the inverse matrices for all the diagonal block of A when (A->nc)>1, and NULL when (A->nc)=1

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 263 of file smoother_str.c.

10.85.2.4 fasp_smoother_dstr_gs_ascend()

```
void fasp_smoother_dstr_gs_ascend (
    dSTRmat * A,
    dvector * b,
    dvector * u,
    REAL * diaginv )
```

Gauss-Seidel method as the smoother in the ascending manner.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
diaginv	All the inverse matrices for all the diagonal block of A when (A->nc)>1, and NULL when (A->nc)=1

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 308 of file smoother_str.c.

10.85.2.5 fasp_smoother_dstr_gs_cf()

```
void fasp_smoother_dstr_gs_cf (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv,
            INT * mark,
            const INT order )
```

Gauss method as the smoother in the C-F manner.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
diaginv	All the inverse matrices for all the diagonal block of A when (A->nc)>1, and NULL when (A->nc)=1
mark	Pointer to the user-defined order array
order	Flag to indicate the order for smoothing CPFIRST 1 : C-points first and then F-points FPFIRST -1 : F-points first and then C-points

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 663 of file smoother_str.c.

10.85.2.6 fasp_smoother_dstr_gs_descend()

```
void fasp_smoother_dstr_gs_descend (
            dSTRmat * A,
```

```
dvector * b,
dvector * u,
REAL * diaginv )
```

Gauss-Seidel method as the smoother in the descending manner.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
diaginv	All the inverse matrices for all the diagonal block of A when (A->nc)>1, and NULL when (A->nc)=1

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 423 of file smoother_str.c.

10.85.2.7 fasp_smoother_dstr_gs_order()

```
void fasp_smoother_dstr_gs_order (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv,
            INT * mark )
```

Gauss method as the smoother in the user-defined order.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
diaginv	All the inverse matrices for all the diagonal block of A when (A->nc)>1, and NULL when (A->nc)=1
mark	Pointer to the user-defined order array

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 540 of file smoother_str.c.

10.85.2.8 fasp_smoother_dstr_jacobi()

```
void fasp_smoother_dstr_jacobi (
            dSTRmat * A,
            dvector * b,
            dvector * u )
```

Jacobi method as the smoother.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 31 of file smoother_str.c.

10.85.2.9 fasp_smoother_dstr_jacobi1()

```
void fasp_smoother_dstr_jacobi1 (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv )
```

Jacobi method as the smoother with diag_inv given.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
diaginv	All the inverse matrices for all the diagonal block of A when (A->nc)>1, and NULL when (A->nc)=1

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 79 of file smoother_str.c.

10.85.2.10 fasp_smoother_dstr_schwarz()

```
void fasp_smoother_dstr_schwarz (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            dvector * diaginv,
            ivector * pivot,
            ivector * neigh,
            ivector * order )
```

Schwarz method as the smoother.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
diaginv	Pointer to dvector: the inverse of the diagonals
pivot	Pointer to ivector: the pivot of diagonal blocks
neigh	Pointer to ivector: neighborhoods
order	Pointer to ivector: the smoothing order

Author

Xiaozhe Hu

Date

10/01/2011

Definition at line 1643 of file smoother_str.c.

10.85.2.11 fasp_smoother_dstr_sor()

```
void fasp_smoother_dstr_sor (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            const INT order,
            INT * mark,
            const REAL weight )
```

SOR method as the smoother.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
order	Flag to indicate the order for smoothing If mark = NULL ASCEND 12: in ascending manner DESCEND 21: in descending manner If mark != NULL USERDEFINED 0 : in the user-defined manner CPFIRST 1 : C-points first and then F-points FPFIRST -1 : F-points first and then C-points
mark	Pointer to the user-defined ordering(when order=0) or CF_marker array(when order!=0)
weight	Over-relaxation weight

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 855 of file smoother_str.c.

10.85.2.12 fasp_smoother_dstr_sor1()

```
void fasp_smoother_dstr_sor1 (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            const INT order,
            INT * mark,
            REAL * diaginv,
            const REAL weight )
```

SOR method as the smoother.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
order	Flag to indicate the order for smoothing If mark = NULL ASCEND 12: in ascending manner DESCEND 21: in descending manner If mark != NULL USERDEFINED 0 : in the user-defined manner CPFIRST 1 : C-points first and then F-points FPFIRST -1 : F-points first and then C-points
mark	Pointer to the user-defined ordering(when order=0) or CF_marker array(when order!=0)
diaginv	Inverse of the diagonal entries
weight	Over-relaxation weight

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 916 of file smoother_str.c.

10.85.2.13 fasp_smoother_dstr_sor_ascend()

```
void fasp_smoother_dstr_sor_ascend (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv,
            REAL weight )
```

SOR method as the smoother in the ascending manner.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
diaginv	All the inverse matrices for all the diagonal block of A when (A->nc)>1, and NULL when (A->nc)=1
weight	Over-relaxation weight

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 962 of file smoother_str.c.

10.85.2.14 fasp_smoother_dstr_sor_cf()

```
void fasp_smoother_dstr_sor_cf (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv,
```

```
INT * mark,
const INT order,
const REAL weight )
```

SOR method as the smoother in the C-F manner.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
diaginv	All the inverse matrices for all the diagonal block of A when (A->nc)>1, and NULL when (A->nc)=1
mark	Pointer to the user-defined order array
order	Flag to indicate the order for smoothing CPFIRST 1 : C-points first and then F-points FPFIRST -1 : F-points first and then C-points
weight	Over-relaxation weight

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 1334 of file smoother_str.c.

10.85.2.15 fasp_smoother_dstr_sor_descend()

```
void fasp_smoother_dstr_sor_descend (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv,
            REAL weight )
```

SOR method as the smoother in the descending manner.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
diaginv	All the inverse matrices for all the diagonal block of A when (A->nc)>1, and NULL when (A->nc)=1
weight	Over-relaxation weight

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 1082 of file smoother_str.c.

10.85.2.16 fasp_smoother_dstr_sor_order()

```
void fasp_smoother_dstr_sor_order (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv,
            INT * mark,
            REAL weight )
```

SOR method as the smoother in the user-defined order.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
diaginv	All the inverse matrices for all the diagonal block of A when (A->nc)>1, and NULL when (A->nc)=1
mark	Pointer to the user-defined order array
weight	Over-relaxation weight

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 1203 of file smoother_str.c.

10.86 sparse_block.c File Reference

Sparse matrix block operations.

```
#include <time.h>
#include "fasp.h"
#include "fasp_block.h"
#include "fasp_functs.h"
```

Functions

- void fasp_dblc_free (dBLCmat ∗A)
  Free block CSR sparse matrix data memory space.
- SHORT fasp_dcsr_getblk (dCSRmat ∗A, INT ∗Is, INT ∗Js, const INT m, const INT n, dCSRmat ∗B)
  Get a sub CSR matrix of A with specified rows and columns.
- SHORT fasp_dbsr_getblk (dBSRmat ∗A, INT ∗Is, INT ∗Js, const INT m, const INT n, dBSRmat ∗B)
  Get a sub BSR matrix of A with specified rows and columns.
- dCSRmat fasp_dbsr_getblk_dcsr (dBSRmat ∗A)
  get dCSRmat block from a dBSRmat matrix
- dCSRmat fasp_dbsr_Linfinity_dcsr (dBSRmat ∗A)
  get dCSRmat from a dBSRmat matrix using L_infinity norm of each small block

10.86.1 Detailed Description

Sparse matrix block operations.

10.86.2 Function Documentation

10.86.2.1 fasp_dblc_free()

```
void fasp_dblc_free (
            dBLCmat * A )
```

Free block CSR sparse matrix data memory space.

Parameters

A	Pointer to the dBLCmat matrix

Author

Xiaozhe Hu

Date

04/18/2014

Definition at line 30 of file sparse_block.c.

10.86.2.2 fasp_dbsr_getblk()

```
SHORT fasp_dbsr_getblk (
            dBSRmat * A,
```

```
INT * Is,
INT * Js,
const INT m,
const INT n,
dBSRmat * B )
```

Get a sub BSR matrix of A with specified rows and columns.

Parameters

A	Pointer to dBSRmat BSR matrix
B	Pointer to dBSRmat BSR matrix
Is	Pointer to selected rows
Js	Pointer to selected columns
m	Number of selected rows
n	Number of selected columns

Returns

FASP_SUCCESS if succeeded, otherwise return error information.

Author

Shiquan Zhang, Xiaozhe Hu

Date

12/25/2010

Modified by Chunsheng Feng, Xiaoqiang Yue on 05/23/2012

Definition at line 160 of file sparse_block.c.

10.86.2.3 fasp_dbsr_getblk_dcsr()

```
dCSRmat fasp_dbsr_getblk_dcsr (
        dBSRmat * A )
```

get dCSRmat block from a dBSRmat matrix

Parameters

$*A$	Pointer to the BSR format matrix

Returns

dCSRmat matrix if succeed, NULL if fail

Author

Xiaozhe Hu

Date

03/16/2012

Definition at line 256 of file sparse_block.c.

10.86.2.4 fasp_dbsr_Linfinity_dcsr()

```
dCSRmat fasp_dbsr_Linfinity_dcsr (
            dBSRmat * A )
```

get dCSRmat from a dBSRmat matrix using L_infinity norm of each small block

Parameters

*A	Pointer to the BSR format matrix

Returns

dCSRmat matrix if succeed, NULL if fail

Author

Xiaozhe Hu

Date

05/25/2014

Definition at line 312 of file sparse_block.c.

10.86.2.5 fasp_dcsr_getblk()

```
SHORT fasp_dcsr_getblk (
            dCSRmat * A,
            INT * Is,
            INT * Js,
            const INT m,
            const INT n,
            dCSRmat * B )
```

Get a sub CSR matrix of A with specified rows and columns.

Parameters

A	Pointer to dCSRmat matrix
B	Pointer to dCSRmat matrix
Is	Pointer to selected rows
Js	Pointer to selected columns
m	Number of selected rows
n	Number of selected columns

Returns

FASP_SUCCESS if succeeded, otherwise return error information.

Author

Shiquan Zhang, Xiaozhe Hu

Date

12/25/2010

Modified by Chunsheng Feng, Xiaoqiang Yue on 05/23/2012

Definition at line 66 of file sparse_block.c.

10.87 sparse_bsr.c File Reference

Sparse matrix operations for dBSRmat matrices.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- dBSRmat fasp_dbsr_create (const INT ROW, const INT COL, const INT NNZ, const INT nb, const INT storage↩_manner)

    Create BSR sparse matrix data memory space.
- void fasp_dbsr_alloc (const INT ROW, const INT COL, const INT NNZ, const INT nb, const INT storage_manner, dBSRmat ∗A)

    Allocate memory space for BSR format sparse matrix.
- void fasp_dbsr_free (dBSRmat ∗A)

    Free memory space for BSR format sparse matrix.
- void fasp_dbsr_null (dBSRmat ∗A)

Initialize sparse matrix on structured grid.
- void fasp_dbsr_cp (dBSRmat ∗A, dBSRmat ∗B)

  copy a dCSRmat to a new one B=A
- INT fasp_dbsr_trans (dBSRmat ∗A, dBSRmat ∗AT)

  Find A^T from given dBSRmat matrix A.
- SHORT fasp_dbsr_diagpref (dBSRmat ∗A)

  Reorder the column and data arrays of a square BSR matrix, so that the first entry in each row is the diagonal one.
- dvector fasp_dbsr_getdiaginv (dBSRmat ∗A)

  Get D^{-1} of matrix A.
- dBSRmat fasp_dbsr_diaginv (dBSRmat ∗A)

  Compute B := D^{-1}∗A, where 'D' is the block diagonal part of A.
- dBSRmat fasp_dbsr_diaginv2 (dBSRmat ∗A, REAL ∗diaginv)

  Compute B := D^{-1}∗A, where 'D' is the block diagonal part of A.
- dBSRmat fasp_dbsr_diaginv3 (dBSRmat ∗A, REAL ∗diaginv)

  Compute B := D^{-1}∗A, where 'D' is the block diagonal part of A.
- dBSRmat fasp_dbsr_diaginv4 (dBSRmat ∗A, REAL ∗diaginv)

  Compute B := D^{-1}∗A, where 'D' is the block diagonal part of A.
- void fasp_dbsr_getdiag (INT n, dBSRmat ∗A, REAL ∗diag)

  Abstract the diagonal blocks of a BSR matrix.
- dBSRmat fasp_dbsr_diagLU (dBSRmat ∗A, REAL ∗DL, REAL ∗DU)

  Compute B := DL∗A∗DU. We decompose each diagonal block of A into LDU form and DL = diag(L^{-1}) and DU = diag(U^{-1}).
- dBSRmat fasp_dbsr_diagLU2 (dBSRmat ∗A, REAL ∗DL, REAL ∗DU)

  Compute B := DL∗A∗DU. We decompose each diagonal block of A into LDU form and DL = diag(L^{-1}) and DU = diag(U^{-1}).
- dBSRmat fasp_dbsr_perm (dBSRmat ∗A, INT ∗P)

  Apply permutation of A, i.e. Aperm=PAP' by the orders given in P.

10.87.1 Detailed Description

Sparse matrix operations for dBSRmat matrices.

10.87.2 Function Documentation

10.87.2.1 fasp_dbsr_alloc()

```
void fasp_dbsr_alloc (
            const INT ROW,
            const INT COL,
            const INT NNZ,
            const INT nb,
            const INT storage_manner,
            dBSRmat * A )
```

Allocate memory space for BSR format sparse matrix.

Parameters

ROW	Number of rows of block
COL	Number of columns of block
NNZ	Number of nonzero blocks
nb	Dimension of each block
storage_manner	Storage manner for each sub-block
A	Pointer to new dBSRmat matrix

Author

Xiaozhe Hu

Date

10/26/2010

Definition at line 87 of file sparse_bsr.c.

10.87.2.2 fasp_dbsr_cp()

```
void fasp_dbsr_cp (
            dBSRmat * A,
            dBSRmat * B )
```

copy a dCSRmat to a new one B=A

Parameters

A	Pointer to the dBSRmat matrix
B	Pointer to the dBSRmat matrix

Author

Xiaozhe Hu

Date

08/07/2011

Definition at line 181 of file sparse_bsr.c.

10.87.2.3 fasp_dbsr_create()

```
dBSRmat fasp_dbsr_create (
            const INT ROW,
            const INT COL,
            const INT NNZ,
            const INT nb,
            const INT storage_manner )
```

Create BSR sparse matrix data memory space.

Parameters

ROW	Number of rows of block
COL	Number of columns of block
NNZ	Number of nonzero blocks
nb	Dimension of each block
storage_manner	Storage manner for each sub-block

Returns

A The new dBSRmat matrix

Author

Xiaozhe Hu

Date

10/26/2010

Definition at line 36 of file sparse_bsr.c.

10.87.2.4 fasp_dbsr_diaginv()

```
dBSRmat fasp_dbsr_diaginv (
            dBSRmat * A )
```

Compute B := D^{-1}*A, where 'D' is the block diagonal part of A.

Parameters

A	Pointer to the dBSRmat matrix

Author

Zhiyang Zhou

Date

2010/10/26

Note

Works for general nb (Xiaozhe)

Modified by Chunsheng Feng, Zheng Li on 08/25/2012

Definition at line 496 of file sparse_bsr.c.

10.87.2.5 fasp_dbsr_diaginv2()

```
dBSRmat fasp_dbsr_diaginv2 (
            dBSRmat * A,
            REAL * diaginv )
```

Compute B := D^{-1}*A, where 'D' is the block diagonal part of A.

Parameters

A	Pointer to the dBSRmat matrix
diaginv	Pointer to the inverses of all the diagonal blocks

Author

Zhiyang Zhou

Date

2010/11/07

Note

Works for general nb (Xiaozhe)

Modified by Chunsheng Feng, Zheng Li on 08/25/2012

Definition at line 660 of file sparse_bsr.c.

10.87.2.6 fasp_dbsr_diaginv3()

```
dBSRmat fasp_dbsr_diaginv3 (
            dBSRmat * A,
            REAL * diaginv )
```

Compute B := D^{-1}*A, where 'D' is the block diagonal part of A.

Parameters

A	Pointer to the dBSRmat matrix
diaginv	Pointer to the inverses of all the diagonal blocks

Returns

BSR matrix after diagonal scaling

Author

Xiaozhe Hu

Date

12/25/2010

Note

Works for general nb (Xiaozhe)

Modified by Xiaozhe Hu on 05/26/2012

Definition at line 762 of file sparse_bsr.c.

10.87.2.7 fasp_dbsr_diaginv4()

```
dBSRmat fasp_dbsr_diaginv4 (
            dBSRmat * A,
            REAL * diaginv )
```

Compute B := D^{-1}*A, where 'D' is the block diagonal part of A.

Parameters

A	Pointer to the dBSRmat matrix
diaginv	Pointer to the inverses of all the diagonal blocks

Returns

BSR matrix after diagonal scaling

Note

Works for general nb (Xiaozhe)
A is pre-ordered that the first block of each row is the diagonal block!

Author

Xiaozhe Hu

Date

03/12/2011

Modified by Chunsheng Feng, Zheng Li on 08/26/2012

Definition at line 1120 of file sparse_bsr.c.

10.87.2.8 fasp_dbsr_diagLU()

```
dBSRmat fasp_dbsr_diagLU (
            dBSRmat * A,
            REAL * DL,
            REAL * DU )
```

Compute B := DL∗A∗DU. We decompose each diagonal block of A into LDU form and DL = diag(L^{-1}) and DU = diag(U^{-1}).

Parameters

A	Pointer to the dBSRmat matrix
DL	Pointer to the diag(L^{-1})
DU	Pointer to the diag(U^{-1})

Returns

BSR matrix after scaling

Author

Xiaozhe Hu

Date

04/02/2014

Definition at line 1449 of file sparse_bsr.c.

10.87.2.9 fasp_dbsr_diagLU2()

```
dBSRmat fasp_dbsr_diagLU2 (
            dBSRmat * A,
            REAL * DL,
            REAL * DU )
```

Compute B := DL*A*DU. We decompose each diagonal block of A into LDU form and DL = diag(L^{-1}) and DU = diag(U^{-1}).

Parameters

A	Pointer to the dBSRmat matrix
DL	Pointer to the diag(L^{-1})
DU	Pointer to the diag(U^{-1})

Returns

BSR matrix after scaling

Author

Zheng Li, Xiaozhe Hu

Date

06/17/2014

Definition at line 1677 of file sparse_bsr.c.

10.87.2.10 fasp_dbsr_diagpref()

```
SHORT fasp_dbsr_diagpref (
            dBSRmat * A )
```

Reorder the column and data arrays of a square BSR matrix, so that the first entry in each row is the diagonal one.

Parameters

A	Pointer to the BSR matrix

Author

Xiaozhe Hu

Date

03/10/2011

Author

Chunsheng Feng, Zheng Li

Date

09/02/2012

Note

Reordering is done in place.

Definition at line 292 of file sparse_bsr.c.

10.87.2.11 fasp_dbsr_free()

```
void fasp_dbsr_free (
            dBSRmat * A )
```

Free memory space for BSR format sparse matrix.

Parameters

A	Pointer to the dBSRmat matrix

Author

Xiaozhe Hu

Date

10/26/2010

Definition at line 133 of file sparse_bsr.c.

10.87.2.12 fasp_dbsr_getdiag()

```
fasp_dbsr_getdiag (
            INT n,
            dBSRmat * A,
            REAL * diag )
```

Abstract the diagonal blocks of a BSR matrix.

Parameters

n	Number of blocks to get
A	Pointer to the 'dBSRmat' type matrix
diag	Pointer to array which stores the diagonal blocks in row by row manner

Author

Zhiyang Zhou

Date

2010/10/26

Note

Works for general nb (Xiaozhe)

Modified by Chunsheng Feng, Zheng Li on 08/25/2012

Definition at line 1411 of file sparse_bsr.c.

10.87.2.13 fasp_dbsr_getdiaginv()

```
dvector fasp_dbsr_getdiaginv (
            dBSRmat * A )
```

Get D^{-1} of matrix A.

Parameters

A	Pointer to the dBSRmat matrix

Author

Xiaozhe Hu

Date

02/19/2013

Note

Works for general nb (Xiaozhe)

Definition at line 392 of file sparse_bsr.c.

10.87.2.14 fasp_dbsr_null()

```
void fasp_dbsr_null (
            dBSRmat * A )
```

Initialize sparse matrix on structured grid.

Parameters

A	Pointer to the dBSRmat matrix

Author

Xiaozhe Hu

Date

10/26/2010

Definition at line 158 of file sparse_bsr.c.

10.87.2.15 fasp_dbsr_perm()

```
dBSRmat fasp_dbsr_perm (
            dBSRmat * A,
            INT * P )
```

Apply permutation of A, i.e. Aperm=PAP' by the orders given in P.

Parameters

A	Pointer to the original dCSRmat matrix
P	Pointer to the given ordering

Returns

The new ordered dCSRmat matrix if succeed, NULL if fail

Author

Zheng Li

Date

24/9/2015

Note

P[i] = k means k-th row and column become i-th row and column!

Definition at line 1878 of file sparse_bsr.c.

10.87.2.16 fasp_dbsr_trans()

```
INT fasp_dbsr_trans (
            dBSRmat * A,
            dBSRmat * AT )
```

Find A^T from given dBSRmat matrix A.

Parameters

A	Pointer to the dBSRmat matrix
AT	Pointer to the transpose of dBSRmat matrix A

Author

Chunsheng FENG

Date

2011/06/08

Modified by Xiaozhe Hu (08/06/2011)

Definition at line 208 of file sparse_bsr.c.

10.88 sparse_coo.c File Reference

Sparse matrix operations for dCOOmat matrices.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- dCOOmat fasp_dcoo_create (const INT m, const INT n, const INT nnz)
  Create IJ sparse matrix data memory space.
- void fasp_dcoo_alloc (const INT m, const INT n, const INT nnz, dCOOmat ∗A)
  Allocate COO sparse matrix memory space.
- void fasp_dcoo_free (dCOOmat ∗A)
  Free IJ sparse matrix data memory space.
- void fasp_dcoo_shift (dCOOmat ∗A, const INT offset)
  Re-index a REAL matrix in IJ format to make the index starting from 0 or 1.

10.88.1 Detailed Description

Sparse matrix operations for dCOOmat matrices.

10.88.2 Function Documentation

10.88.2.1 fasp_dcoo_alloc()

```
void fasp_dcoo_alloc (
            const INT m,
            const INT n,
            const INT nnz,
            dCOOmat * A )
```

Allocate COO sparse matrix memory space.

Parameters

m	Number of rows
n	Number of columns
nnz	Number of nonzeros
A	Pointer to the dCSRmat matrix

Author

Xiaozhe Hu

Date

03/25/2013

Definition at line 62 of file sparse_coo.c.

10.88.2.2 fasp_dcoo_create()

```
dCOOmat fasp_dcoo_create (
            const INT m,
            const INT n,
            const INT nnz )
```

Create IJ sparse matrix data memory space.

Parameters

m	Number of rows
n	Number of columns
nnz	Number of nonzeros

Returns

A The new dCOOmat matrix

Author

Chensong Zhang

Date

2010/04/06

Definition at line 34 of file sparse_coo.c.

10.88.2.3 fasp_dcoo_free()

```
void fasp_dcoo_free (
            dCOOmat * A )
```

Free IJ sparse matrix data memory space.

Parameters

A	Pointer to the dCOOmat matrix

Author

Chensong Zhang

Date

2010/04/03

Definition at line 94 of file sparse_coo.c.

10.88.2.4 fasp_dcoo_shift()

```
void fasp_dcoo_shift (
            dCOOmat * A,
            const INT offset )
```

Re-index a REAL matrix in IJ format to make the index starting from 0 or 1.

Parameters

A	Pointer to IJ matrix
offset	Size of offset (1 or -1)

Author

Chensong Zhang

Date

2010/04/06

Modified by Chunsheng Feng, Zheng Li on 08/25/2012

Definition at line 116 of file sparse_coo.c.

10.89 sparse_csr.c File Reference

Sparse matrix operations for dCSRmat matrices.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- dCSRmat fasp_dcsr_create (const INT m, const INT n, const INT nnz)

    Create CSR sparse matrix data memory space.
- iCSRmat fasp_icsr_create (const INT m, const INT n, const INT nnz)

    Create CSR sparse matrix data memory space.
- void fasp_dcsr_alloc (const INT m, const INT n, const INT nnz, dCSRmat ∗A)

    Allocate CSR sparse matrix memory space.
- void fasp_dcsr_free (dCSRmat ∗A)

    Free CSR sparse matrix data memory space.
- void fasp_icsr_free (iCSRmat ∗A)

    Free CSR sparse matrix data memory space.
- void fasp_dcsr_null (dCSRmat ∗A)

    Initialize CSR sparse matrix.
- void fasp_icsr_null (iCSRmat ∗A)

    Initialize CSR sparse matrix.
- dCSRmat fasp_dcsr_perm (dCSRmat ∗A, INT ∗P)

    Apply permutation of A, i.e. Aperm=PAP' by the orders given in P.
- void fasp_dcsr_sort (dCSRmat ∗A)

    Sort each row of A in ascending order w.r.t. column indices.
- void fasp_dcsr_getdiag (INT n, dCSRmat ∗A, dvector ∗diag)

    Get first n diagonal entries of a CSR matrix A.
- void fasp_dcsr_getcol (const INT n, dCSRmat ∗A, REAL ∗col)

    Get the n-th column of a CSR matrix A.
- void fasp_dcsr_diagpref (dCSRmat ∗A)

    Re-order the column and data arrays of a CSR matrix, so that the first entry in each row is the diagonal.
- SHORT fasp_dcsr_regdiag (dCSRmat ∗A, REAL value)

    Regularize diagonal entries of a CSR sparse matrix.
- void fasp_icsr_cp (iCSRmat ∗A, iCSRmat ∗B)

    Copy a iCSRmat to a new one B=A.
- void fasp_dcsr_cp (dCSRmat ∗A, dCSRmat ∗B)

    copy a dCSRmat to a new one B=A
- void fasp_icsr_trans (iCSRmat ∗A, iCSRmat ∗AT)

    Find transpose of iCSRmat matrix A.
- INT fasp_dcsr_trans (dCSRmat ∗A, dCSRmat ∗AT)

    Find transpose of dCSRmat matrix A.
- void **fasp_dcsr_transpose** (INT ∗row[2], INT ∗col[2], REAL ∗val[2], INT ∗nn, INT ∗tniz)
- void fasp_dcsr_compress (dCSRmat ∗A, dCSRmat ∗B, REAL dtol)

    Compress a CSR matrix A and store in CSR matrix B by dropping small entries abs(aij)<=dtol.
- SHORT fasp_dcsr_compress_inplace (dCSRmat ∗A, REAL dtol)

    Compress a CSR matrix A IN PLACE by dropping small entries abs(aij)<=dtol.
- void fasp_dcsr_shift (dCSRmat ∗A, INT offset)

    Re-index a REAL matrix in CSR format to make the index starting from 0 or 1.
- void fasp_dcsr_symdiagscale (dCSRmat ∗A, dvector ∗diag)

    Symmetric diagonal scaling D^{-1/2}AD^{-1/2}.
- dCSRmat fasp_dcsr_sympat (dCSRmat ∗A)

    Get symmetric part of a dCSRmat matrix.

- void fasp_dcsr_multicoloring (dCSRmat ∗A, INT ∗flags, INT ∗groups)

  Use the greedy multi-coloring to get color groups of the adjacency graph of A.
- void fasp_dcsr_transz (dCSRmat ∗A, INT ∗p, dCSRmat ∗AT)

  Generalized transpose of A: (n x m) matrix given in dCSRmat format.
- dCSRmat fasp_dcsr_permz (dCSRmat ∗A, INT ∗p)

  Permute rows and cols of A, i.e. A=PAP' by the ordering in p.
- void fasp_dcsr_sortz (dCSRmat ∗A, const SHORT isym)

  Sort each row of A in ascending order w.r.t. column indices.

10.89.1 Detailed Description

Sparse matrix operations for dCSRmat matrices.

10.89.2 Function Documentation

10.89.2.1 fasp_dcsr_alloc()

```
void fasp_dcsr_alloc (
            const INT m,
            const INT n,
            const INT nnz,
            dCSRmat * A )
```

Allocate CSR sparse matrix memory space.

Parameters

m	Number of rows
n	Number of columns
nnz	Number of nonzeros
A	Pointer to the dCSRmat matrix

Author

Chensong Zhang

Date

2010/04/06

Definition at line 125 of file sparse_csr.c.

10.89.2.2 fasp_dcsr_compress()

```
void fasp_dcsr_compress (
            dCSRmat * A,
            dCSRmat * B,
            REAL dtol )
```

Compress a CSR matrix A and store in CSR matrix B by dropping small entries abs(aij)<=dtol.

Parameters

A	Pointer to dCSRmat CSR matrix
B	Pointer to dCSRmat CSR matrix
dtol	Drop tolerance

Author

Shiquan Zhang

Date

03/10/2010

Modified by Chunsheng Feng, Zheng Li on 08/25/2012

Definition at line 957 of file sparse_csr.c.

10.89.2.3 fasp_dcsr_compress_inplace()

```
SHORT fasp_dcsr_compress_inplace (
            dCSRmat * A,
            REAL dtol )
```

Compress a CSR matrix A IN PLACE by dropping small entries abs(aij)<=dtol.

Parameters

A	Pointer to dCSRmat CSR matrix
dtol	Drop tolerance

Author

Xiaozhe Hu

Date

12/25/2010

Modified by Chensong Zhang on 02/21/2013

Note

This routine can be modified for filtering.

Definition at line 1037 of file sparse_csr.c.

10.89.2.4 fasp_dcsr_cp()

```
void fasp_dcsr_cp (
            dCSRmat * A,
            dCSRmat * B )
```

copy a dCSRmat to a new one B=A

Parameters

A	Pointer to the dCSRmat matrix
B	Pointer to the dCSRmat matrix

Author

Chensong Zhang

Date

04/06/2010

Modified by Chunsheng Feng, Xiaoqiang Yue on 05/23/2012

Definition at line 723 of file sparse_csr.c.

10.89.2.5 fasp_dcsr_create()

```
dCSRmat fasp_dcsr_create (
            const INT m,
            const INT n,
            const INT nnz )
```

Create CSR sparse matrix data memory space.

Parameters

m	Number of rows
n	Number of columns
nnz	Number of nonzeros

Returns

A the new dCSRmat matrix

Author

Chensong Zhang

Date

2010/04/06

Definition at line 34 of file sparse_csr.c.

10.89.2.6 fasp_dcsr_diagpref()

```
void fasp_dcsr_diagpref (
            dCSRmat * A )
```

Re-order the column and data arrays of a CSR matrix, so that the first entry in each row is the diagonal.

Parameters

A	Pointer to the matrix to be re-ordered

Author

Zhiyang Zhou

Date

09/09/2010

Author

Chunsheng Feng, Zheng Li

Date

09/02/2012

Note

Reordering is done in place.

Modified by Chensong Zhang on Dec/21/2012

Definition at line 553 of file sparse_csr.c.

10.89.2.7 fasp_dcsr_free()

```
void fasp_dcsr_free (
            dCSRmat * A )
```

Free CSR sparse matrix data memory space.

Parameters

A	Pointer to the dCSRmat matrix

Author

Chensong Zhang

Date

2010/04/06

Definition at line 166 of file sparse_csr.c.

10.89.2.8 fasp_dcsr_getcol()

```
void fasp_dcsr_getcol (
            const INT n,
            dCSRmat * A,
            REAL * col )
```

Get the n-th column of a CSR matrix A.

Parameters

n	Index of a column of A (0 <= n <= A.col-1)
A	Pointer to dCSRmat CSR matrix
col	Pointer to the column

Author

Xiaozhe Hu

Date

11/07/2009

Modified by Chunsheng Feng, Zheng Li on 07/08/2012

Definition at line 474 of file sparse_csr.c.

10.89.2.9 fasp_dcsr_getdiag()

```
void fasp_dcsr_getdiag (
            INT n,
            dCSRmat * A,
            dvector * diag )
```

Get first n diagonal entries of a CSR matrix A.

Parameters

n	Number of diagonal entries to get (if n=0, then get all diagonal entries)
A	Pointer to dCSRmat CSR matrix
diag	Pointer to the diagonal as a dvector

Author

Chensong Zhang

Date

05/20/2009

Modified by Chunsheng Feng, Xiaoqiang Yue on 05/23/2012

Definition at line 410 of file sparse_csr.c.

10.89.2.10 fasp_dcsr_multicoloring()

```
void fasp_dcsr_multicoloring (
            dCSRmat * A,
            INT * flags,
            INT * groups )
```

Use the greedy multi-coloring to get color groups of the adjacency graph of A.

Parameters

A	Input dCSRmat
flags	flags for the independent group
groups	Return group numbers

Author

Chunsheng Feng

Date

09/15/2012

Definition at line 1265 of file sparse_csr.c.

10.89.2.11 fasp_dcsr_null()

```
void fasp_dcsr_null (
            dCSRmat * A )
```

Initialize CSR sparse matrix.

Parameters

A	Pointer to the dCSRmat matrix

Author

Chensong Zhang

Date

2010/04/03

Definition at line 204 of file sparse_csr.c.

10.89.2.12 fasp_dcsr_perm()

```
dCSRmat fasp_dcsr_perm (
            dCSRmat * A,
            INT * P )
```

Apply permutation of A, i.e. Aperm=PAP' by the orders given in P.

Parameters

A	Pointer to the original dCSRmat matrix
P	Pointer to orders

Returns

The new ordered dCSRmat matrix if succeed, NULL if fail

Author

Shiquan Zhang

Date

03/10/2010

Note

P[i] = k means k-th row and column become i-th row and column!
Deprecated! Will be replaced by fasp_dcsr_permz later. –Chensong

Modified by Chunsheng Feng, Zheng Li on 07/12/2012

Definition at line 247 of file sparse_csr.c.

10.89.2.13 fasp_dcsr_permz()

```
dCSRmat fasp_dcsr_permz (
            dCSRmat * A,
            INT * p )
```

Permute rows and cols of A, i.e. A=PAP' by the ordering in p.

Parameters

A	Pointer to the original dCSRmat matrix
p	Pointer to ordering

Note

This is just applying twice fasp_dcsr_transz(&A,p,At).
In matlab notation: Aperm=A(p,p);

Returns

The new ordered dCSRmat matrix if succeed, NULL if fail

Author

Ludmil Zikatanov

Date

19951219 (Fortran), 20150912 (C)

Definition at line 1486 of file sparse_csr.c.

10.89.2.14 fasp_dcsr_regdiag()

```
SHORT fasp_dcsr_regdiag (
            dCSRmat * A,
            REAL value )
```

Regularize diagonal entries of a CSR sparse matrix.

Parameters

A	Pointer to the dCSRmat matrix
value	Set a value on diag(A) which is too close to zero to "value"

Returns

FASP_SUCCESS if no diagonal entry is close to zero, else ERROR

Author

Shiquan Zhang

Date

11/07/2009

Definition at line 659 of file sparse_csr.c.

10.89.2.15 fasp_dcsr_shift()

```
void fasp_dcsr_shift (
            dCSRmat * A,
            INT offset )
```

Re-index a REAL matrix in CSR format to make the index starting from 0 or 1.

Parameters

A	Pointer to CSR matrix
offset	Size of offset (1 or -1)

Author

Chensong Zhang

Date

04/06/2010

Modified by Chunsheng Feng, Zheng Li on 07/11/2012

Definition at line 1085 of file sparse_csr.c.

10.89.2.16 fasp_dcsr_sort()

```
void fasp_dcsr_sort (
            dCSRmat * A )
```

Sort each row of A in ascending order w.r.t. column indices.

Parameters

A	Pointer to the dCSRmat matrix

Author

Shiquan Zhang

Date

06/10/2010

Definition at line 358 of file sparse_csr.c.

10.89.2.17 fasp_dcsr_sortz()

```
void fasp_dcsr_sortz (
            dCSRmat * A,
            const SHORT isym )
```

Sort each row of A in ascending order w.r.t. column indices.

Parameters

A	Pointer to the dCSRmat matrix
isym	Flag for symmetry, =[0/nonzero]=[general/symmetric] matrix

Note

Applying twice fasp_dcsr_transz(), if A is symmetric, then the transpose is applied only once and then AT copied on A.

Author

Ludmil Zikatanov

Date

19951219 (Fortran), 20150912 (C)

Definition at line 1518 of file sparse_csr.c.

10.89.2.18 fasp_dcsr_symdiagscale()

```
void fasp_dcsr_symdiagscale (
            dCSRmat * A,
            dvector * diag )
```

Symmetric diagonal scaling D^{-1/2}AD^{-1/2}.

Parameters

A	Pointer to the dCSRmat matrix
diag	Pointer to the diagonal entries

Author

Xiaozhe Hu

Date

01/31/2011

Modified by Chunsheng Feng, Zheng Li on 07/11/2012

Definition at line 1146 of file sparse_csr.c.

10.89.2.19 fasp_dcsr_sympat()

```
dCSRmat fasp_dcsr_sympat (
            dCSRmat * A )
```

Get symmetric part of a dCSRmat matrix.

Parameters

$*A$	pointer to the dCSRmat matrix

Returns

symmetrized the dCSRmat matrix

Author

Xiaozhe Hu

Date

03/21/2011

Definition at line 1232 of file sparse_csr.c.

10.89.2.20 fasp_dcsr_trans()

```
void fasp_dcsr_trans (
            dCSRmat * A,
            dCSRmat * AT )
```

Find transpose of dCSRmat matrix A.

Parameters

A	Pointer to the dCSRmat matrix
AT	Pointer to the transpose of dCSRmat matrix A (output)

Author

Chensong Zhang

Date

04/06/2010

Modified by Chunsheng Feng, Zheng Li on 06/20/2012

Definition at line 826 of file sparse_csr.c.

10.89.2.21 fasp_dcsr_transz()

```
void fasp_dcsr_transz (
            dCSRmat * A,
            INT * p,
            dCSRmat * AT )
```

Generalized transpose of A: (n x m) matrix given in dCSRmat format.

Parameters

A	Pointer to matrix in dCSRmat for transpose, INPUT
p	Permutation, INPUT
AT	Pointer to matrix AT = transpose(A) if p = NULL, OR AT = transpose(A)p if p is not NULL

Note

The storage for all pointers in AT should already be allocated, i.e. AT->IA, AT->JA and AT->val should be allocated before calling this function. If A.val=NULL, then AT->val[] is not changed.
performs AT=transpose(A)p, where p is a permutation. If p=NULL then p=I is assumed. Applying twice this procedure one gets At=transpose(transpose(A)p)p = transpose(p)Ap, which is the same A with rows and columns permutted according to p.
If A=NULL, then only transposes/permutes the structure of A.
For p=NULL, applying this two times A–>AT–>A orders all the row indices in A in increasing order.

Reference: Fred G. Gustavson. Two fast algorithms for sparse matrices: multiplication and permuted transposition. ACM Trans. Math. Software, 4(3):250–269, 1978.

Author

Ludmil Zikatanov

Date

19951219 (Fortran), 20150912 (C)

Definition at line 1366 of file sparse_csr.c.

10.89.2.22 fasp_icsr_cp()

```
void fasp_icsr_cp (
            iCSRmat * A,
            iCSRmat * B )
```

Copy a iCSRmat to a new one B=A.

Parameters

A	Pointer to the iCSRmat matrix
B	Pointer to the iCSRmat matrix

Author

Chensong Zhang

Date

05/16/2013

Definition at line 698 of file sparse_csr.c.

10.89.2.23 fasp_icsr_create()

```
iCSRmat fasp_icsr_create (
            const INT m,
            const INT n,
            const INT nnz )
```

Create CSR sparse matrix data memory space.

Parameters

m	Number of rows
n	Number of columns
nnz	Number of nonzeros

Returns

A the new iCSRmat matrix

Author

Chensong Zhang

Date

2010/04/06

Definition at line 80 of file sparse_csr.c.

10.89.2.24 fasp_icsr_free()

```
void fasp_icsr_free (
            iCSRmat * A )
```

Free CSR sparse matrix data memory space.

Parameters

A	Pointer to the iCSRmat matrix

Author

Chensong Zhang

Date

2010/04/06

Definition at line 185 of file sparse_csr.c.

10.89.2.25 fasp_icsr_null()

```
void fasp_icsr_null (
            iCSRmat * A )
```

Initialize CSR sparse matrix.

Parameters

A	Pointer to the iCSRmat matrix

Author

Chensong Zhang

Date

2010/04/03

Definition at line 221 of file sparse_csr.c.

10.89.2.26 fasp_icsr_trans()

```
void fasp_icsr_trans (
            iCSRmat * A,
            iCSRmat * AT )
```

Find transpose of iCSRmat matrix A.

Parameters

A	Pointer to the iCSRmat matrix A
AT	Pointer to the iCSRmat matrix A'

Returns

The transpose of iCSRmat matrix A

Author

Chensong Zhang

Date

04/06/2010

Modified by Chunsheng Feng, Zheng Li on 06/20/2012

Definition at line 750 of file sparse_csr.c.

10.90 sparse_csrl.c File Reference

Sparse matrix operations for dCSRLmat matrices.

```
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- dCSRLmat ∗ fasp_dcsrl_create (const INT num_rows, const INT num_cols, const INT num_nonzeros)

  Create a dCSRLmat object.
- void fasp_dcsrl_free (dCSRLmat ∗A)

  Destroy a dCSRLmat object.

10.90.1 Detailed Description

Sparse matrix operations for dCSRLmat matrices.

Note

For details of CSRL format, refer to Optimizing sparse matrix vector product computations using unroll and jam by John Mellor-Crummey and John Garvin, Tech Report Rice Univ, Aug 2002.

10.90.2 Function Documentation

10.90.2.1 fasp_dcsrl_create()

```
dCSRLmat * fasp_dcsrl_create (
            const INT num_rows,
            const INT num_cols,
            const INT num_nonzeros )
```

Create a dCSRLmat object.

Parameters

num_rows	Number of rows
num_cols	Number of cols
num_nonzeros	Number of nonzero entries

Author

Zhiyang Zhou

Date

01/07/2001

Definition at line 30 of file sparse_csrl.c.

10.90.2.2 fasp_dcsrl_free()

```
void fasp_dcsrl_free (
            dCSRLmat * A )
```

Destroy a dCSRLmat object.

Parameters

A	Pointer to the dCSRLmat type matrix

Author

Zhiyang Zhou

Date

01/07/2011

Definition at line 58 of file sparse_csrl.c.

10.91 sparse_str.c File Reference

Sparse matrix operations for dSTRmat matrices.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_dstr_null (dSTRmat ∗A)

    Initialize sparse matrix on structured grid.
- dSTRmat fasp_dstr_create (const INT nx, const INT ny, const INT nz, const INT nc, const INT nband, INT ∗offsets)

    Create STR sparse matrix data memory space.
- void fasp_dstr_alloc (const INT nx, const INT ny, const INT nz, const INT nxy, const INT ngrid, const INT nband, const INT nc, INT ∗offsets, dSTRmat ∗A)

    Allocate STR sparse matrix memory space.
- void fasp_dstr_free (dSTRmat ∗A)

    Free STR sparse matrix data memeory space.
- void fasp_dstr_cp (dSTRmat ∗A, dSTRmat ∗B)

    Copy a dSTRmat to a new one B=A.

10.91.1 Detailed Description

Sparse matrix operations for dSTRmat matrices.

10.91.2 Function Documentation

10.91.2.1 fasp_dstr_alloc()

```
void fasp_dstr_alloc (
            const INT nx,
            const INT ny,
            const INT nz,
            const INT nxy,
            const INT ngrid,
            const INT nband,
            const INT nc,
            INT * offsets,
            dSTRmat * A )
```

Allocate STR sparse matrix memory space.

Parameters

nx	Number of grids in x direction
ny	Number of grids in y direction
nz	Number of grids in z direction
nxy	Number of grids in x-y plane
ngrid	Number of grids
nband	Number of off-diagonal bands
nc	Number of components
offsets	Shift from diagonal
A	Pointer to the dSTRmat matrix

Author

Shiquan Zhang, Xiaozhe Hu

Date

05/17/2010

Definition at line 109 of file sparse_str.c.

10.91.2.2 fasp_dstr_cp()

```
void fasp_dstr_cp (
            dSTRmat * A,
            dSTRmat * B )
```

Copy a dSTRmat to a new one B=A.

Parameters

A	Pointer to the dSTRmat matrix
B	Pointer to the dSTRmat matrix

Author

Zhiyang Zhou

Date

04/21/2010

Definition at line 181 of file sparse_str.c.

10.91.2.3 fasp_dstr_create()

```
dSTRmat fasp_dstr_create (
            const INT nx,
            const INT ny,
            const INT nz,
            const INT nc,
            const INT nband,
            INT * offsets )
```

Create STR sparse matrix data memory space.

Parameters

nx	Number of grids in x direction
ny	Number of grids in y direction
nz	Number of grids in z direction
nc	Number of components
nband	Number of off-diagonal bands
offsets	Shift from diagonal

Returns

The dSTRmat matrix

Author

Shiquan Zhang, Xiaozhe Hu

Date

05/17/2010

Definition at line 57 of file sparse_str.c.

10.91.2.4 fasp_dstr_free()

```
void fasp_dstr_free (
            dSTRmat * A )
```

Free STR sparse matrix data memeory space.

Parameters

A	Pointer to the dSTRmat matrix

Author

Shiquan Zhang, Xiaozhe Hu

Date

05/17/2010

Definition at line 152 of file sparse_str.c.

10.91.2.5 fasp_dstr_null()

```
void fasp_dstr_null (
            dSTRmat * A )
```

Initialize sparse matrix on structured grid.

Parameters

A	Pointer to the dSTRmat matrix

Author

Shiquan Zhang, Xiaozhe Hu

Date

05/17/2010

Definition at line 25 of file sparse_str.c.

10.92 sparse_util.c File Reference

Routines for sparse matrix operations.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- void fasp_sparse_abybms_ (INT ∗ia, INT ∗ja, INT ∗ib, INT ∗jb, INT ∗nap, INT ∗map, INT ∗mbp, INT ∗ic, INT ∗jc)

    Multiplication of two sparse matrices: calculating the nonzero structure of the result if jc is not null. If jc is null only finds num of nonzeroes.
- void fasp_sparse_abyb_ (INT ∗ia, INT ∗ja, REAL ∗a, INT ∗ib, INT ∗jb, REAL ∗b, INT ∗nap, INT ∗map, INT ∗mbp, INT ∗ic, INT ∗jc, REAL ∗c)

    Multiplication of two sparse matrices: calculating the numerical values in the result.
- void fasp_sparse_iit_ (INT ∗ia, INT ∗ja, INT ∗na, INT ∗ma, INT ∗iat, INT ∗jat)

    Transpose a boolean matrix (only given by ia, ja)
- void fasp_sparse_aat_ (INT ∗ia, INT ∗ja, REAL ∗a, INT ∗na, INT ∗ma, INT ∗iat, INT ∗jat, REAL ∗at)

    transpose a boolean matrix (only given by ia, ja)
- void fasp_sparse_aplbms_ (INT ∗ia, INT ∗ja, INT ∗ib, INT ∗jb, INT ∗nab, INT ∗mab, INT ∗ic, INT ∗jc)

    Addition of two sparse matrices: calculating the nonzero structure of the result if jc is not null. if jc is null only finds num of nonzeroes.

- void fasp_sparse_aplusb_ (INT ∗ia, INT ∗ja, REAL ∗a, INT ∗ib, INT ∗jb, REAL ∗b, INT ∗nab, INT ∗mab, INT ∗ic, INT ∗jc, REAL ∗c)

  Addition of two sparse matrices: calculating the numerical values in the result.
- void fasp_sparse_rapms_ (INT ∗ir, INT ∗jr, INT ∗ia, INT ∗ja, INT ∗ip, INT ∗jp, INT ∗nin, INT ∗ncin, INT ∗iac, INT ∗jac, INT ∗maxrout)

  Calculates the nonzero structure of R∗A∗P, if jac is not null. If jac is null only finds num of nonzeroes.
- void fasp_sparse_wtams_ (INT ∗jw, INT ∗ia, INT ∗ja, INT ∗nwp, INT ∗map, INT ∗jv, INT ∗nvp, INT ∗icp)

  Finds the nonzeroes in the result of v^t = w^t A, where w is a sparse vector and A is sparse matrix. jv is an integer array containing the indices of the nonzero elements in the result.
- void fasp_sparse_wta_ (INT ∗jw, REAL ∗w, INT ∗ia, INT ∗ja, REAL ∗a, INT ∗nwp, INT ∗map, INT ∗jv, REAL ∗v, INT ∗nvp)

  Calculate v^t = w^t A, where w is a sparse vector and A is sparse matrix. v is an array of dimension = number of columns in A.
- void fasp_sparse_ytxbig_ (INT ∗jy, REAL ∗y, INT ∗nyp, REAL ∗x, REAL ∗s)

  Calculates s = y^t x. y-sparse, x - no.
- void fasp_sparse_ytx_ (INT ∗jy, REAL ∗y, INT ∗jx, REAL ∗x, INT ∗nyp, INT ∗nxp, INT ∗icp, REAL ∗s)

  Calculates s = y^t x. y is sparse, x is sparse.
- void fasp_sparse_rapcmp_ (INT ∗ir, INT ∗jr, REAL ∗r, INT ∗ia, INT ∗ja, REAL ∗a, INT ∗ipt, INT ∗jpt, REAL ∗pt, INT ∗nin, INT ∗ncin, INT ∗iac, INT ∗jac, REAL ∗ac, INT ∗idummy)

  Calculates R∗A∗P after the nonzero structure of the result is known. iac,jac,ac have to be allocated before call to this function.
- ivector fasp_sparse_MIS (dCSRmat ∗A)

  get the maximal independet set of a CSR matrix

10.92.1 Detailed Description

Routines for sparse matrix operations.

Note

Most algorithms work as follows: (a) Boolean operations (to determine the nonzero structure); (b) Numerical part, where the result is calculated.
: Parameter notation :I: is input; :O: is output; :IO: is both
C-version: by Ludmil Zikatanov 2010-04-08 tested 2010-04-08

: Modifed Xiaozhe Hu 2010-10-18

Todo Remove unwanted functions from this file. –Chensong

10.92.2 Function Documentation

10.92.2.1 fasp_sparse_aat_()

```
void fasp_sparse_aat_ (
            INT * ia,
            INT * ja,
            REAL * a,
```

```
            INT * na,
            INT * ma,
            INT * iat,
            INT * jat,
            REAL * at )
```

transpose a boolean matrix (only given by ia, ja)

Parameters

ia	array of row pointers (as usual in CSR)
ja	array of column indices
a	array of entries of teh input
na	number of rows of A
ma	number of cols of A
iat	array of row pointers in the result
jat	array of column indices
at	array of entries of the result

Definition at line 272 of file sparse_util.c.

10.92.2.2 fasp_sparse_abyb_()

```
void fasp_sparse_abyb_ (
            INT * ia,
            INT * ja,
            REAL * a,
            INT * ib,
            INT * jb,
            REAL * b,
            INT * nap,
            INT * map,
            INT * mbp,
            INT * ic,
            INT * jc,
            REAL * c )
```

Multiplication of two sparse matrices: calculating the numerical values in the result.

Parameters

ia	array of row pointers 1st multiplicand
ja	array of column indices 1st multiplicand
a	entries of the 1st multiplicand
ib	array of row pointers 2nd multiplicand
jb	array of column indices 2nd multiplicand
b	entries of the 2nd multiplicand
ic	array of row pointers in c=a*b

Parameters

jc	array of column indices in c=a∗b
c	entries of the result: c= a∗b
nap	number of rows in the 1st multiplicand
map	number of columns in the 1st multiplicand
mbp	number of columns in the 2nd multiplicand

Modified by Chensong Zhang on 09/11/2012

Definition at line 124 of file sparse_util.c.

10.92.2.3 fasp_sparse_abybms_()

```
void fasp_sparse_abybms_ (
            INT * ia,
            INT * ja,
            INT * ib,
            INT * jb,
            INT * nap,
            INT * map,
            INT * mbp,
            INT * ic,
            INT * jc )
```

Multiplication of two sparse matrices: calculating the nonzero structure of the result if jc is not null. If jc is null only finds num of nonzeroes.

Parameters

ia	array of row pointers 1st multiplicand
ia	array of row pointers 1st multiplicand
ja	array of column indices 1st multiplicand
ib	array of row pointers 2nd multiplicand
jb	array of column indices 2nd multiplicand
nap	number of rows of A
map	number of cols of A
mbp	number of cols of b
ic	array of row pointers in the result (this is also computed here again, so that we can have a stand alone call of this routine, if for some reason the number of nonzeros in the result is known)
jc	array of column indices in the result c=a∗b

Modified by Chensong Zhang on 09/11/2012

Definition at line 53 of file sparse_util.c.

10.92.2.4 fasp_sparse_aplbms_()

```
void void fasp_sparse_aplbms_ (
            INT * ia,
            INT * ja,
            INT * ib,
            INT * jb,
            INT * nab,
            INT * mab,
            INT * ic,
            INT * jc )
```

Addition of two sparse matrices: calculating the nonzero structure of the result if jc is not null. if jc is null only finds num of nonzeroes.

Parameters

ia	array of row pointers 1st summand
ia	array of row pointers 1st summand
ja	array of column indices 1st summand
ib	array of row pointers 2nd summand
jb	array of column indices 2nd summand
nab	number of rows
mab	number of cols
ic	array of row pointers in the result (this is also computed here again, so that we can have a stand alone call of this routine, if for some reason the number of nonzeros in the result is known)
jc	array of column indices in the result c=a+b

Definition at line 359 of file sparse_util.c.

10.92.2.5 fasp_sparse_aplusb_()

```
void fasp_sparse_aplusb_ (
            INT * ia,
            INT * ja,
            REAL * a,
            INT * ib,
            INT * jb,
            REAL * b,
            INT * nab,
            INT * mab,
            INT * ic,
            INT * jc,
            REAL * c )
```

Addition of two sparse matrices: calculating the numerical values in the result.

Parameters

ia	array of row pointers 1st summand
ja	array of column indices 1st summand
a	entries of the 1st summand
ib	array of row pointers 2nd summand
jb	array of column indices 2nd summand
b	entries of the 2nd summand
nab	number of rows
mab	number of cols
ic	array of row pointers in c=a+b
jc	array of column indices in c=a+b
c	entries of the result: c=a+b

Definition at line 431 of file sparse_util.c.

10.92.2.6 fasp_sparse_iit_()

```
void fasp_sparse_iit_ (
            INT * ia,
            INT * ja,
            INT * na,
            INT * ma,
            INT * iat,
            INT * jat )
```

Transpose a boolean matrix (only given by ia, ja)

Parameters

ia	array of row pointers (as usual in CSR)
ja	array of column indices
na	number of rows
ma	number of cols
iat	array of row pointers in the result
jat	array of column indices

Note

For the concrete algorithm, see:

Definition at line 197 of file sparse_util.c.

10.92.2.7 fasp_sparse_MIS()

```
ivector fasp_sparse_MIS (
            dCSRmat * A )
```

get the maximal independet set of a CSR matrix

Parameters

A	pointer to the matrix

Note

: only use the sparsity of A, index starts from 1 (fortran)!!

Definition at line 909 of file sparse_util.c.

10.92.2.8 fasp_sparse_rapcmp_()

```
void fasp_sparse_rapcmp_ (
            INT * ir,
            INT * jr,
            REAL * r,
            INT * ia,
            INT * ja,
            REAL * a,
            INT * ipt,
            INT * jpt,
            REAL * pt,
            INT * nin,
            INT * ncin,
            INT * iac,
            INT * jac,
            REAL * ac,
            INT * idummy )
```

Calculates R∗A∗P after the nonzero structure of the result is known. iac,jac,ac have to be allocated before call to this function.

Note

:I: is input :O: is output :IO: is both

Parameters

ir	:I: array of row pointers for R
jr	:I: array of column indices for R
r	:I: entries of R
ia	:I: array of row pointers for A
ja	:I: array of column indices for A
a	:I: entries of A
ipt	:I: array of row pointers for P
jpt	:I: array of column indices for P
pt	:I: entries of P

Parameters

nin	:I: number of rows in R
ncin	:I: number of rows in
iac	:O: array of row pointers for P
jac	:O: array of column indices for P
ac	:O: entries of P
idummy	not changed

Note

compute R∗A∗P for known nonzero structure of the result the result is stored in iac,jac,ac!

Definition at line 788 of file sparse_util.c.

10.92.2.9 fasp_sparse_rapms_()

```
void fasp_sparse_rapms_ (
            INT * ir,
            INT * jr,
            INT * ia,
            INT * ja,
            INT * ip,
            INT * jp,
            INT * nin,
            INT * ncin,
            INT * iac,
            INT * jac,
            INT * maxrout )
```

Calculates the nonzero structure of R∗A∗P, if jac is not null. If jac is null only finds num of nonzeroes.

Note

:I: is input :O: is output :IO: is both

Parameters

ir	:I: array of row pointers for R
jr	:I: array of column indices for R
ia	:I: array of row pointers for A
ja	:I: array of column indices for A
ip	:I: array of row pointers for P
jp	:I: array of column indices for P
nin	:I: number of rows in R
ncin	:I: number of columns in R
iac	:O: array of row pointers for Ac
jac	:O: array of column indices for Ac
maxrout	:O: the maximum nonzeroes per row for R

Note

Computes the sparsity pattern of R∗A∗P. maxrout is output and is the maximum nonzeroes per row for r. On output we also have is iac (if jac is null) and jac (if jac entry is not null). R is (nc,n) A is (n,n) and P is (n,nc)!

Modified by Chensong Zhang on 09/11/2012

Definition at line 514 of file sparse_util.c.

10.92.2.10 fasp_sparse_wta_()

```
void fasp_sparse_wta_ (
            INT * jw,
            REAL * w,
            INT * ia,
            INT * ja,
            REAL * a,
            INT * nwp,
            INT * map,
            INT * jv,
            REAL * v,
            INT * nvp )
```

Calculate v^t = w^t A, where w is a sparse vector and A is sparse matrix. v is an array of dimension = number of columns in A.

Note

:I: is input :O: is output :IO: is both

Parameters

jw	:I: indices such that w[jw] is nonzero
w	:I: the values of w
ia	:I: array of row pointers for A
ja	:I: array of column indices for A
a	:I: entries of A
nwp	:I: number of nonzeroes in w (the length of w)
map	:I: number of columns in A
jv	:O: indices such that v[jv] is nonzero
v	:O: the result v^t=w^t A
nvp	:I: number of nonzeroes in v

Definition at line 648 of file sparse_util.c.

10.92.2.11 fasp_sparse_wtams_()

```
void fasp_sparse_wtams_ (
```

```
              INT * jw,
              INT * ia,
              INT * ja,
              INT * nwp,
              INT * map,
              INT * jv,
              INT * nvp,
              INT * icp )
```

Finds the nonzeroes in the result of v^t = w^t A, where w is a sparse vector and A is sparse matrix. jv is an integer array containing the indices of the nonzero elements in the result.

:I: is input :O: is output :IO: is both

Parameters

jw	:I: indices such that w[jw] is nonzero
ia	:I: array of row pointers for A
ja	:I: array of column indices for A
nwp	:I: number of nonzeroes in w (the length of w)
map	:I: number of columns in A
jv	:O: indices such that v[jv] is nonzero
nvp	:I: number of nonzeroes in v
icp	:IO: is a working array of length (*map) which on output satisfies icp[jv[k]-1]=k; Values of icp[] at positions * other than (jv[k]-1) remain unchanged.

Modified by Chensong Zhang on 09/11/2012

Definition at line 595 of file sparse_util.c.

10.92.2.12 fasp_sparse_ytx_()

```
void fasp_sparse_ytx_ (
              INT * jy,
              REAL * y,
              INT * jx,
              REAL * x,
              INT * nyp,
              INT * nxp,
              INT * icp,
              REAL * s )
```

Calculates s = y^t x. y is sparse, x is sparse.

note :I: is input :O: is output :IO: is both

Parameters

jy	:I: indices such that y[jy] is nonzero
y	:I: is a sparse vector.

Parameters

nyp	:I: number of nonzeroes in y
jx	:I: indices such that x[jx] is nonzero
x	:I: is a sparse vector.
nxp	:I: number of nonzeroes in x
icp	???
s	:O: s = y^t x.

Definition at line 733 of file sparse_util.c.

10.92.2.13 fasp_sparse_ytxbig_()

```
void fasp_sparse_ytxbig_ (
            INT * jy,
            REAL * y,
            INT * nyp,
            REAL * x,
            REAL * s )
```

Calculates s = y^t x. y-sparse, x - no.

Note

:I: is input :O: is output :IO: is both

Parameters

jy	:I: indices such that y[jy] is nonzero
y	:I: is a sparse vector.
nyp	:I: number of nonzeroes in v
x	:I: also a vector assumed to have entry for any j=jy[i]-1; for i=1:nyp. This means that x here does not have to be sparse.
s	:O: s = y^t x.

Definition at line 699 of file sparse_util.c.

10.93 spbcgs.c File Reference

Krylov subspace methods – Preconditioned BiCGstab with safety net.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_dcsr_spbcgs (dCSRmat *A, dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned BiCGstab method for solving Au=b with safety net.
- INT fasp_solver_dbsr_spbcgs (dBSRmat *A, dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned BiCGstab method for solving Au=b with safety net.
- INT fasp_solver_dblc_spbcgs (dBLCmat *A, dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned BiCGstab method for solving Au=b with safety net.
- INT fasp_solver_dstr_spbcgs (dSTRmat *A, dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned BiCGstab method for solving Au=b with safety net.

10.93.1 Detailed Description

Krylov subspace methods – Preconditioned BiCGstab with safety net.

Abstract algorithm

PBICGStab method to solve A*x=b is to generate {x_k} to approximate x

Note: We generate a series of {p_k} such that V_k=span{p_1,...,p_k}.

Step 0. Given A, b, x_0, M

Step 1. Compute residual r_0 = b-A*x_0 and convergence check;

Step 2. Initialization z_0 = M^{-1}*r_0, p_0=z_0;

Step 3. Main loop ...

FOR k = 0:MaxIt

- get step size alpha = f(r_k,z_k,p_k);
- update solution: x_{k+1} = x_k + alpha*p_k;
- check whether x is NAN;
- perform stagnation check;
- update residual: r_{k+1} = r_k - alpha*(A*p_k);
- if r_{k+1} < r_{best}: save x_{k+1} as x_{best};
- perform residual check;
- obtain p_{k+1} using {p_0, p_1, ... , p_k};
- prepare for next iteration;
- print the result of k-th iteration; END FOR

Convergence check: norm(r)/norm(b) < tol

Stagnation check:

- IF norm(alpha*p_k)/norm(x_{k+1}) < tol_stag
    1. compute r=b-A*x_{k+1};
    2. convergence check;
    3. IF (not converged & restart_number < Max_Stag_Check) restart;
- END IF

Residual check:

- IF norm(r_{k+1})/norm(b) < tol
    1. compute the real residual r = b-A*x_{k+1};
    2. convergence check;
    3. IF (not converged & restart_number < Max_Res_Check) restart;
- END IF

safety net check:

- IF r_{k+1} > r_{best}
    1. x_{k+1} = x_{best}
- END IF

Note

Refer to Y. Saad 2003 Iterative methods for sparse linear systems (2nd Edition), SIAM
See spbcgs.c for a safer version

10.93.2 Function Documentation

10.93.2.1 fasp_solver_dblc_spbcgs()

```
INT fasp_solver_dblc_spbcgs (
            dBLCmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned BiCGstab method for solving Au=b with safety net.

Parameters

A	Pointer to dBLCmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to the structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

03/31/2013

Definition at line 868 of file spbcgs.c.

10.93.2.2 fasp_solver_dbsr_spbcgs()

```
INT fasp_solver_dbsr_spbcgs (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned BiCGstab method for solving Au=b with safety net.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to the structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

03/31/2013

Definition at line 479 of file spbcgs.c.

10.93.2.3 fasp_solver_dcsr_spbcgs()

```
INT fasp_solver_dcsr_spbcgs (
            dCSRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned BiCGstab method for solving Au=b with safety net.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to the structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

03/31/2013

Definition at line 90 of file spbcgs.c.

10.93.2.4 fasp_solver_dstr_spbcgs()

```
INT fasp_solver_dstr_spbcgs (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned BiCGstab method for solving Au=b with safety net.

Parameters

A	Pointer to dSTRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to the structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

03/31/2013

Definition at line 1257 of file spbcgs.c.

10.94 spcg.c File Reference

Krylov subspace methods – Preconditioned conjugate gradient with safety net.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_dcsr_spcg (dCSRmat *A, dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

    Preconditioned conjugate gradient method for solving Au=b with safety net.
- INT fasp_solver_dblc_spcg (dBLCmat *A, dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

    Preconditioned conjugate gradient method for solving Au=b with safety net.
- INT fasp_solver_dstr_spcg (dSTRmat *A, dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

    Preconditioned conjugate gradient method for solving Au=b with safety net.

10.94.1 Detailed Description

Krylov subspace methods – Preconditioned conjugate gradient with safety net.

Abstract algorithm

PCG method to solve A*x=b is to generate {x_k} to approximate x

Step 0. Given A, b, x_0, M

Step 1. Compute residual r_0 = b-A*x_0 and convergence check;

Step 2. Initialization z_0 = M^{-1}*r_0, p_0=z_0;

Step 3. Main loop ...

FOR k = 0:MaxIt

- get step size alpha = f(r_k,z_k,p_k);
- update solution: x_{k+1} = x_k + alpha*p_k;
- check whether x is NAN;
- perform stagnation check;
- update residual: r_{k+1} = r_k - alpha*(A*p_k);
- if r_{k+1} < r_{best}: save x_{k+1} as x_{best};

- perform residual check;
- obtain p_{k+1} using {p_0, p_1, ... , p_k};
- prepare for next iteration;
- print the result of k-th iteration; END FOR

Convergence check: norm(r)/norm(b) < tol

Stagnation check:

- IF norm(alpha∗p_k)/norm(x_{k+1}) < tol_stag
  1. compute r=b-A∗x_{k+1};
  2. convergence check;
  3. IF (not converged & restart_number < Max_Stag_Check) restart;
- END IF

Residual check:

- IF norm(r_{k+1})/norm(b) < tol
  1. compute the real residual r = b-A∗x_{k+1};
  2. convergence check;
  3. IF (not converged & restart_number < Max_Res_Check) restart;
- END IF

safety net check:

- IF r_{k+1} > r_{best}
  1. x_{k+1} = x_{best}
- END IF

Note

Refer to Y. Saad 2003 Iterative methods for sparse linear systems (2nd Edition), SIAM
See pcg.c for a version without safety net

10.94.2 Function Documentation

10.94.2.1 fasp_solver_dblc_spcg()

```
INT fasp_solver_dblc_spcg (
            dBLCmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned conjugate gradient method for solving Au=b with safety net.

Parameters

A	Pointer to dBLCmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to the structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

03/28/2013

Definition at line 419 of file spcg.c.

10.94.2.2 fasp_solver_dcsr_spcg()

```
INT fasp_solver_dcsr_spcg (
            dCSRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned conjugate gradient method for solving Au=b with safety net.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to the structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

03/28/2013

Definition at line 88 of file spcg.c.

10.94.2.3 fasp_solver_dstr_spcg()

```
INT fasp_solver_dstr_spcg (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned conjugate gradient method for solving Au=b with safety net.

Parameters

A	Pointer to dSTRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
MaxIt	Maximal number of iterations
tol	Tolerance for stopping
pc	Pointer to the structure of precondition (precond)
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

03/28/2013

Definition at line 750 of file spcg.c.

10.95 spgmres.c File Reference

Krylov subspace methods – Preconditioned GMRes with safety net.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_dcsr_spgmres (dCSRmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, const REAL tol, const INT MaxIt, SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned GMRES method for solving Au=b with safe-guard.
- INT fasp_solver_dblc_spgmres (dBLCmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, const REAL tol, const INT MaxIt, SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned GMRES method for solving Au=b with safe-guard.
- INT fasp_solver_dbsr_spgmres (dBSRmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, const REAL tol, const INT MaxIt, SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned GMRES method for solving Au=b with safe-guard.
- INT fasp_solver_dstr_spgmres (dSTRmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, const REAL tol, const INT MaxIt, SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned GMRES method for solving Au=b with safe-guard.

10.95.1 Detailed Description

Krylov subspace methods – Preconditioned GMRes with safety net.

Note

Refer to Y. Saad 2003 Iterative methods for sparse linear systems (2nd Edition), SIAM
See also pgmres.c for a variable restarting version.
See pgmres.c for a version without safety net

10.95.2 Function Documentation

10.95.2.1 fasp_solver_dblc_spgmres()

```
INT fasp_solver_dblc_spgmres (
            dBLCmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned GMRES method for solving Au=b with safe-guard.

Parameters

A	Pointer to dBLCmat: the coefficient matrix
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns
pc	Pointer to the structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/05/2013

Definition at line 386 of file spgmres.c.

10.95.2.2 fasp_solver_dbsr_spgmres()

```
INT fasp_solver_dbsr_spgmres (
            dBSRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned GMRES method for solving Au=b with safe-guard.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns
pc	Pointer to the structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/05/2013

Definition at line 726 of file spgmres.c.

10.95.2.3 fasp_solver_dcsr_spgmres()

```
INT fasp_solver_dcsr_spgmres (
            dCSRmat * A,
            dvector * b,
            dvector * x,
```

```
precond * pc,
const REAL tol,
const INT MaxIt,
SHORT restart,
const SHORT stop_type,
const SHORT prtlvl )
```

Preconditioned GMRES method for solving Au=b with safe-guard.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns
pc	Pointer to the structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/05/2013 Modified by Chunsheng Feng on 07/22/2013: Add adapt memory allocate

Definition at line 46 of file spgmres.c.

10.95.2.4 fasp_solver_dstr_spgmres()

```
INT fasp_solver_dstr_spgmres (
            dSTRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned GMRES method for solving Au=b with safe-guard.

Parameters

A	Pointer to dSTRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns
pc	Pointer to the structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/05/2013

Definition at line 1066 of file spgmres.c.

10.96 spminres.c File Reference

Krylov subspace methods – Preconditioned minimal residual with safety net.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_dcsr_spminres (dCSRmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

    A preconditioned minimal residual (Minres) method for solving Au=b with safety net.
- INT fasp_solver_dblc_spminres (dBLCmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

    A preconditioned minimal residual (Minres) method for solving Au=b with safety net.
- INT fasp_solver_dstr_spminres (dSTRmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

    A preconditioned minimal residual (Minres) method for solving Au=b with safety net.

10.96.1 Detailed Description

Krylov subspace methods – Preconditioned minimal residual with safety net.

Abstract algorithm

Krylov method to solve A∗x=b is to generate {x_k} to approximate x, where x_k is the optimal solution in Krylov space

V_k=span{r_0,A∗r_0,A^2∗r_0,...,A^{k-1}∗r_0},

under some inner product.

For the implementation, we generate a series of {p_k} such that V_k=span{p_1,...,p_k}. Details:

Step 0. Given A, b, x_0, M

Step 1. Compute residual r_0 = b-A∗x_0 and convergence check;

Step 2. Initialization z_0 = M^{-1}∗r_0, p_0=z_0;

Step 3. Main loop ...

FOR k = 0:MaxIt

- get step size alpha = f(r_k,z_k,p_k);
- update solution: x_{k+1} = x_k + alpha∗p_k;
- check whether x is NAN;
- perform stagnation check;
- update residual: r_{k+1} = r_k - alpha∗(A∗p_k);
- if r_{k+1} < r_{best}: save x_{k+1} as x_{best};
- perform residual check;
- obtain p_{k+1} using {p_0, p_1, ... , p_k};
- prepare for next iteration;
- print the result of k-th iteration; END FOR

Convergence check: norm(r)/norm(b) < tol

Stagnation check:

- IF norm(alpha∗p_k)/norm(x_{k+1}) < tol_stag
  1. compute r=b-A∗x_{k+1};
  2. convergence check;
  3. IF (not converged & restart_number < Max_Stag_Check) restart;
- END IF

Residual check:

- IF norm(r_{k+1})/norm(b) < tol
    1. compute the real residual r = b-A*x_{k+1};
    2. convergence check;
    3. IF (not converged & restart_number < Max_Res_Check) restart;
- END IF

safety net check:

- IF r_{k+1} > r_{best}
    1. x_{k+1} = x_{best}
- END IF

Note

Refer to Y. Saad 2003 Iterative methods for sparse linear systems (2nd Edition), SIAM
See pminres.c for a version without safety net

10.96.2 Function Documentation

10.96.2.1 fasp_solver_dblc_spminres()

```
INT fasp_solver_dblc_spminres (
            dBLCmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

A preconditioned minimal residual (Minres) method for solving Au=b with safety net.

Parameters

A	Pointer to dBLCmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to the structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/09/2013

Definition at line 544 of file spminres.c.

10.96.2.2 fasp_solver_dcsr_spminres()

```
INT fasp_solver_dcsr_spminres (
            dCSRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

A preconditioned minimal residual (Minres) method for solving Au=b with safety net.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to the structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/09/2013

Definition at line 95 of file spminres.c.

10.96.2.3 fasp_solver_dstr_spminres()

```
INT fasp_solver_dstr_spminres (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

A preconditioned minimal residual (Minres) method for solving Au=b with safety net.

Parameters

A	Pointer to dSTRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
MaxIt	Maximal number of iterations
tol	Tolerance for stopping
pc	Pointer to the structure of precondition (precond)
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/09/2013

Definition at line 993 of file spminres.c.

10.97 spvgmres.c File Reference

Krylov subspace methods – Preconditioned variable-restart GMRes with safety net.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
```

Functions

- INT fasp_solver_dcsr_spvgmres (dCSRmat *A, dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Solve "Ax=b" using PGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during the iteration.
- INT fasp_solver_dblc_spvgmres (dBLCmat *A, dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned GMRES method for solving Au=b.
- INT fasp_solver_dbsr_spvgmres (dBSRmat *A, dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Solve "Ax=b" using PGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during the iteration.
- INT fasp_solver_dstr_spvgmres (dSTRmat *A, dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Solve "Ax=b" using PGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during the iteration.

10.97.1 Detailed Description

Krylov subspace methods – Preconditioned variable-restart GMRes with safety net.

Note

Refer to A.H. Baker, E.R. Jessup, and Tz.V. Kolev A Simple Strategy for Varying the Restart Parameter in GMR←ES(m) Journal of Computational and Applied Mathematics, 230 (2009) pp. 751-761. UCRL-JRNL-235266.
See pvgmres.c a version without safety net

10.97.2 Function Documentation

10.97.2.1 fasp_solver_dblc_spvgmres()

```
INT fasp_solver_dblc_spvgmres (
            dBLCmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
```

```
const REAL tol,
const INT MaxIt,
SHORT restart,
const SHORT stop_type,
const SHORT prtlvl )
```

Preconditioned GMRES method for solving Au=b.

Parameters

A	Pointer to dBLCmat: the coefficient matrix
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns
pc	Pointer to the structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/06/2013

Definition at line 425 of file spvgmres.c.

10.97.2.2 fasp_solver_dbsr_spvgmres()

```
INT fasp_solver_dbsr_spvgmres (
            dBSRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Solve "Ax=b" using PGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during the iteration.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns
pc	Pointer to the structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/06/2013

Definition at line 803 of file spvgmres.c.

10.97.2.3 fasp_solver_dcsr_spvgmres()

```
INT fasp_solver_dcsr_spvgmres (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Solve "Ax=b" using PGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during the iteration.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns
pc	Pointer to the structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/06/2013 Modified by Chunsheng Feng on 07/22/2013: Add adapt memory allocate

Definition at line 48 of file spvgmres.c.

10.97.2.4 fasp_solver_dstr_spvgmres()

```
INT fasp_solver_dstr_spvgmres (
            dSTRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Solve "Ax=b" using PGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during the iteration.

Parameters

A	Pointer to dSTRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns
pc	Pointer to the structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/06/2013

Definition at line 1181 of file spvgmres.c.

10.98 threads.c File Reference

Get and set number of threads and assign work load for each thread.

```
#include <stdio.h>
#include <stdlib.h>
#include "fasp.h"
```

Functions

- void FASP_GET_START_END (INT procid, INT nprocs, INT n, INT ∗start, INT ∗end)

  Assign Load to each thread.
- void fasp_set_GS_threads (INT mythreads, INT its)

  Set threads for CPR. Please add it at the begin of Krylov OpenMP method function and after iter++.

Variables

- INT THDs_AMG_GS =0
- INT THDs_CPR_lGS =0
- INT THDs_CPR_gGS =0

10.98.1 Detailed Description

Get and set number of threads and assign work load for each thread.

10.98.2 Function Documentation

10.98.2.1 FASP_GET_START_END()

```
void FASP_GET_START_END (
            INT procid,
            INT nprocs,
            INT n,
            INT * start,
            INT * end )
```

Assign Load to each thread.

Parameters

procid	Index of thread
nprocs	Number of threads
n	Total workload
start	Pointer to the begin of each thread in total workload
end	Pointer to the end of each thread in total workload

Author

Chunsheng Feng, Xiaoqiang Yue and Zheng Li

Date

June/25/2012

Definition at line 83 of file threads.c.

10.98.2.2 fasp_set_GS_threads()

```
void fasp_set_GS_threads (
            INT threads,
            INT its )
```

Set threads for CPR. Please add it at the begin of Krylov OpenMP method function and after iter++.

Parameters

threads	Total threads of solver
its	Current its of the Krylov methods

Author

Feng Chunsheng, Yue Xiaoqiang

Date

03/20/2011

TODO: Why put it here??? –Chensong

Definition at line 125 of file threads.c.

10.98.3 Variable Documentation

10.98.3.1 THDs_AMG_GS

```
INT THDs_AMG_GS =0
```

AMG GS smoothing threads

Definition at line 107 of file threads.c.

10.98.3.2 THDs_CPR_gGS

```
INT THDs_CPR_gGS =0
```

global matrix GS smoothing threads

Definition at line 109 of file threads.c.

10.98.3.3 THDs_CPR_lGS

```
INT THDs_CPR_lGS =0
```

reservoir GS smoothing threads

Definition at line 108 of file threads.c.

10.99 timing.c File Reference

Timing subroutines.

```
#include <time.h>
#include "fasp.h"
```

Functions

- void fasp_gettime (REAL *time)

  Get system time.

10.99.1 Detailed Description

Timing subroutines.

10.99.2 Function Documentation

10.99.2.1 fasp_gettime()

```
fasp_gettime (
            REAL * time )
```

Get system time.

Author

Chunsheng Feng, Zheng LI

Date

11/10/2012

Modified by Chensong Zhang on 09/22/2014: Use CLOCKS_PER_SEC for cross-platform

Definition at line 28 of file timing.c.

10.100 vec.c File Reference

Simple operations for vectors.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
```

Functions

- INT fasp_dvec_isnan (dvector ∗u)

  Check a dvector whether there is NAN.
- dvector fasp_dvec_create (const INT m)

  Create dvector data space of REAL type.
- ivector fasp_ivec_create (const INT m)

  Create vector data space of INT type.
- void fasp_dvec_alloc (const INT m, dvector ∗u)

  Create dvector data space of REAL type.
- void fasp_ivec_alloc (const INT m, ivector ∗u)

  Create vector data space of INT type.
- void fasp_dvec_free (dvector ∗u)

  Free vector data space of REAL type.
- void fasp_ivec_free (ivector ∗u)

  Free vector data space of INT type.
- void fasp_dvec_null (dvector ∗x)

  Initialize dvector.
- void fasp_dvec_rand (const INT n, dvector ∗x)

  Generate random REAL vector in the range from 0 to 1.
- void fasp_dvec_set (INT n, dvector ∗x, REAL val)

  Initialize dvector x[i]=val for i=0:n-1.
- void fasp_ivec_set (const INT m, ivector ∗u)

  Set ivector value to be m.
- void fasp_dvec_cp (dvector ∗x, dvector ∗y)

  Copy dvector x to dvector y.
- REAL fasp_dvec_maxdiff (dvector ∗x, dvector ∗y)

  Maximal difference of two dvector x and y.
- void fasp_dvec_symdiagscale (dvector ∗b, dvector ∗diag)

  Symmetric diagonal scaling D^{-1/2}b.

10.100.1 Detailed Description

Simple operations for vectors.

Note

All structures should be initialized before usage.

10.100.2 Function Documentation

10.100.2.1 fasp_dvec_alloc()

```
void fasp_dvec_alloc (
            const INT m,
            dvector * u )
```

Create dvector data space of REAL type.

Parameters

m	Number of rows
u	Pointer to dvector (OUTPUT)

Author

Chensong Zhang

Date

2010/04/06

Definition at line 99 of file vec.c.

10.100.2.2 fasp_dvec_cp()

```
void fasp_dvec_cp (
            dvector * x,
            dvector * y )
```

Copy dvector x to dvector y.

Parameters

x	Pointer to dvector
y	Pointer to dvector (MODIFIED)

Author

Chensong Zhang

Date

11/16/2009

Definition at line 345 of file vec.c.

10.100.2.3 fasp_dvec_create()

```
dvector fasp_dvec_create (
            const INT m )
```

Create dvector data space of REAL type.

Parameters

m	Number of rows

Returns

u The new dvector

Author

Chensong Zhang

Date

2010/04/06

Definition at line 56 of file vec.c.

10.100.2.4 fasp_dvec_free()

```
void fasp_dvec_free (
            dvector * u )
```

Free vector data space of REAL type.

Parameters

u	Pointer to dvector which needs to be deallocated

Author

Chensong Zhang

Date

2010/04/03

Definition at line 139 of file vec.c.

10.100.2.5 fasp_dvec_isnan()

```
INT fasp_dvec_isnan (
            dvector * u )
```

Check a dvector whether there is NAN.

Parameters

u	Pointer to dvector

Returns

Return TRUE if there is NAN

Author

Chensong Zhang

Date

2013/03/31

Definition at line 33 of file vec.c.

10.100.2.6 fasp_dvec_maxdiff()

```
REAL fasp_dvec_maxdiff (
            dvector * x,
            dvector * y )
```

Maximal difference of two dvector x and y.

Parameters

x	Pointer to dvector
y	Pointer to dvector

Returns

Maximal norm of x-y

Author

Chensong Zhang

Date

11/16/2009

Modified by chunsheng Feng, Zheng Li

Date

06/30/2012

Definition at line 368 of file vec.c.

10.100.2.7 fasp_dvec_null()

```
void fasp_dvec_null (
            dvector * x )
```

Initialize dvector.

Parameters

x	Pointer to dvector which needs to be initialized

Author

Chensong Zhang

Date

2010/04/03

Definition at line 177 of file vec.c.

10.100.2.8 fasp_dvec_rand()

```
void fasp_dvec_rand (
            const INT n,
            dvector * x )
```

Generate random REAL vector in the range from 0 to 1.

Parameters

n	Size of the vector
x	Pointer to dvector

Note

Sample usage:

dvector xapp;

fasp_dvec_create(100,&xapp);

fasp_dvec_rand(100,&xapp);

fasp_dvec_print(100,&xapp);

Author

Chensong Zhang

Date

11/16/2009

Definition at line 203 of file vec.c.

10.100.2.9 fasp_dvec_set()

```
void fasp_dvec_set (
            INT n,
            dvector * x,
            REAL val )
```

Initialize dvector x[i]=val for i=0:n-1.

Parameters

n	Number of variables
x	Pointer to dvector
val	Initial value for the vector

Author

Chensong Zhang

Date

11/16/2009

Modified by Chunsheng Feng, Xiaoqiang Yue on 05/23/2012

Definition at line 235 of file vec.c.

10.100.2.10 fasp_dvec_symdiagscale()

```
void fasp_dvec_symdiagscale (
            dvector * b,
            dvector * diag )
```

Symmetric diagonal scaling D^{-1/2}b.

Parameters

b	Pointer to dvector
diag	Pointer to dvector: the diagonal entries

Author

Xiaozhe Hu

Date

01/31/2011

Definition at line 421 of file vec.c.

10.100.2.11 fasp_ivec_alloc()

```
void fasp_ivec_alloc (
            const INT m,
            ivector * u )
```

Create vector data space of INT type.

Parameters

m	Number of rows
u	Pointer to ivector (OUTPUT)

Author

Chensong Zhang

Date

2010/04/06

Definition at line 119 of file vec.c.

10.100.2.12 fasp_ivec_create()

```
ivector fasp_ivec_create (
            const INT m )
```

Create vector data space of INT type.

Parameters

m	Number of rows

Returns

u The new ivector

Author

Chensong Zhang

Date

2010/04/06

Definition at line 78 of file vec.c.

10.100.2.13 fasp_ivec_free()

```
void fasp_ivec_free (
            ivector * u )
```

Free vector data space of INT type.

Parameters

u	Pointer to ivector which needs to be deallocated

Author

Chensong Zhang

Date

2010/04/03

Note

This function is same as fasp_dvec_free except input type.

Definition at line 159 of file vec.c.

10.100.2.14 fasp_ivec_set()

```
void fasp_ivec_set (
            const INT m,
            ivector * u )
```

Set ivector value to be m.

Parameters

m	Integer value of ivector
u	Pointer to ivector (MODIFIED)

Author

Chensong Zhang

Date

04/03/2010

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

05/23/2012

Definition at line 304 of file vec.c.

10.101 wrapper.c File Reference

Wrappers for accessing functions by advanced users.

```
#include "fasp.h"
#include "fasp_block.h"
#include "fasp_functs.h"
```

Functions

- void fasp_fwrapper_amg_ (INT ∗n, INT ∗nnz, INT ∗ia, INT ∗ja, REAL ∗a, REAL ∗b, REAL ∗u, REAL ∗tol, INT ∗maxit, INT ∗ptrlvl)

  Solve Ax=b by Ruge and Stuben's classic AMG.
- void fasp_fwrapper_krylov_amg_ (INT ∗n, INT ∗nnz, INT ∗ia, INT ∗ja, REAL ∗a, REAL ∗b, REAL ∗u, REAL ∗tol, INT ∗maxit, INT ∗ptrlvl)

  Solve Ax=b by Krylov method preconditioned by classic AMG.
- INT fasp_wrapper_dbsr_krylov_amg (INT n, INT nnz, INT nb, INT ∗ia, INT ∗ja, REAL ∗a, REAL ∗b, REAL ∗u, REAL tol, INT maxit, INT ptrlvl)

  Solve Ax=b by Krylov method preconditioned by AMG (dcsr - > dbsr)
- INT fasp_wrapper_dcoo_dbsr_krylov_amg (INT n, INT nnz, INT nb, INT ∗ia, INT ∗ja, REAL ∗a, REAL ∗b, REAL ∗u, REAL tol, INT maxit, INT ptrlvl)

  Solve Ax=b by Krylov method preconditioned by AMG (dcoo - > dbsr)

10.101.1 Detailed Description

Wrappers for accessing functions by advanced users.

10.101.2 Function Documentation

10.101.2.1 fasp_fwrapper_amg_()

```
void fasp_fwrapper_amg_ (
            INT ∗ n,
            INT ∗ nnz,
            INT ∗ ia,
            INT ∗ ja,
            REAL ∗ a,
            REAL ∗ b,
            REAL ∗ u,
            REAL ∗ tol,
            INT ∗ maxit,
            INT ∗ ptrlvl )
```

Solve Ax=b by Ruge and Stuben's classic AMG.

Parameters

n	Number of cols of A
nnz	Number of nonzeros of A
ia	IA of A in CSR format
ja	JA of A in CSR format
a	VAL of A in CSR format
b	RHS vector
u	Solution vector
tol	Tolerance for iterative solvers
maxit	Max number of iterations
ptrlvl	Print level for iterative solvers

Author

Chensong Zhang

Date

09/16/2010

Definition at line 35 of file wrapper.c.

10.101.2.2 fasp_fwrapper_krylov_amg_()

```
void fasp_fwrapper_krylov_amg_ (
            INT * n,
            INT * nnz,
            INT * ia,
            INT * ja,
            REAL * a,
            REAL * b,
            REAL * u,
            REAL * tol,
            INT * maxit,
            INT * ptrlvl )
```

Solve Ax=b by Krylov method preconditioned by classic AMG.

Parameters

n	Number of cols of A
nnz	Number of nonzeros of A
ia	IA of A in CSR format
ja	JA of A in CSR format
a	VAL of A in CSR format
b	RHS vector
u	Solution vector
tol	Tolerance for iterative solvers
maxit	Max number of iterations
ptrlvl	Print level for iterative solvers

Author

Chensong Zhang

Date

09/16/2010

Definition at line 85 of file wrapper.c.

10.101.2.3 fasp_wrapper_dbsr_krylov_amg()

```
INT fasp_wrapper_dbsr_krylov_amg (
            INT n,
            INT nnz,
            INT nb,
            INT * ia,
            INT * ja,
            REAL * a,
            REAL * b,
            REAL * u,
            REAL tol,
            INT maxit,
            INT ptrlvl )
```

Solve Ax=b by Krylov method preconditioned by AMG (dcsr - > dbsr)

Parameters

n	Number of cols of A
nnz	Number of nonzeros of A
nb	Size of each small block
ia	IA of A in CSR format
ja	JA of A in CSR format
a	VAL of A in CSR format
b	RHS vector
u	Solution vector
tol	Tolerance for iterative solvers
maxit	Max number of iterations
ptrlvl	Print level for iterative solvers

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

03/05/2013

Definition at line 152 of file wrapper.c.

10.101.2.4 fasp_wrapper_dcoo_dbsr_krylov_amg()

```
INT fasp_wrapper_dcoo_dbsr_krylov_amg (
            INT n,
            INT nnz,
            INT nb,
            INT * ia,
            INT * ja,
            REAL * a,
            REAL * b,
            REAL * u,
            REAL tol,
            INT maxit,
            INT ptrlvl )
```

Solve Ax=b by Krylov method preconditioned by AMG (dcoo - $>$ dbsr)

Parameters

n	Number of cols of A
nnz	Number of nonzeros of A
nb	Size of each small block
ia	IA of A in COO format
ja	JA of A in COO format
a	VAL of A in COO format
b	RHS vector
u	Solution vector
tol	Tolerance for iterative solvers
maxit	Max number of iterations
ptrlvl	Print level for iterative solvers

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

03/06/2013

Definition at line 238 of file wrapper.c.

Index

__FASPBLOCK_HEADER__
fasp_block.h, 195
__FASP_HEADER__
fasp.h, 185

A
precond_FASP_blkoil_data, 80
precond_sweeping_data, 85
A_diag
precond_block_data, 66
ABS
fasp.h, 185
AMG_ILU_levels
input_param, 48
AMG_Schwarz_levels
input_param, 50
AMG_aggregation_type
input_param, 47
AMG_aggressive_level
input_param, 47
AMG_aggressive_path
input_param, 47
AMG_amli_degree
input_param, 47
AMG_coarse_dof
input_param, 47
AMG_coarse_scaling
input_param, 48
AMG_coarse_solver
input_param, 48
AMG_coarsening_type
input_param, 48
AMG_cycle_type
input_param, 48
AMG_data, 23
AMG_data_bsr, 24
AMG_interpolation_type
input_param, 48
AMG_levels
input_param, 49
AMG_max_aggregation
input_param, 49
AMG_max_row_sum
input_param, 49
AMG_maxit
input_param, 49
AMG_nl_amli_krylov_type
input_param, 49
AMG_pair_number
input_param, 49
AMG_param, 26
AMG_polynomial_degree
input_param, 50
AMG_postsmooth_iter
input_param, 50
AMG_presmooth_iter
input_param, 50
AMG_quality_bound
input_param, 50
AMG_relaxation
input_param, 50
AMG_smooth_filter
input_param, 51
AMG_smooth_order
input_param, 51
AMG_smoother
input_param, 51
AMG_strong_coupled
input_param, 51
AMG_strong_threshold
input_param, 51
AMG_tentative_smooth
input_param, 51
AMG_tol
input_param, 52
AMG_truncation_threshold
input_param, 52
AMG_type
input_param, 52
AMLI_CYCLE
fasp_const.h, 201
ASCEND
fasp_const.h, 201
Ablc
precond_block_data, 66
Ai
precond_sweeping_data, 85
amg.c, 89
fasp_solver_amg, 89
amg_setup_cr.c, 90
fasp_amg_setup_cr, 91

amg_setup_rs.c, 92
fasp_amg_setup_rs, 92
amg_setup_sa.c, 93
fasp_amg_setup_sa, 93
fasp_amg_setup_sa_bsr, 94
amg_setup_ua.c, 95
fasp_amg_setup_ua, 95
fasp_amg_setup_ua_bsr, 96
amg_solve.c, 96
fasp_amg_solve, 97
fasp_amg_solve_amli, 98
fasp_amg_solve_nl_amli, 98
fasp_famg_solve, 99
amgparam
precond_block_data, 66
amlirecur.c, 100
fasp_amg_amli_coef, 100
fasp_solver_amli, 101
fasp_solver_nl_amli, 101
fasp_solver_nl_amli_bsr, 102
array.c, 103
fasp_array_cp, 104
fasp_array_cp_nc3, 104
fasp_array_cp_nc5, 105
fasp_array_cp_nc7, 105
fasp_array_invpermut_nb, 106
fasp_array_null, 106
fasp_array_permut_nb, 107
fasp_array_set, 108
fasp_iarray_cp, 108
fasp_iarray_set, 109

BIGREAL
fasp_const.h, 201
blas_array.c, 109
fasp_blas_array_ax, 110
fasp_blas_array_axpby, 111
fasp_blas_array_axpy, 111
fasp_blas_array_axpyz, 112
fasp_blas_array_dotprod, 113
fasp_blas_array_norm1, 113
fasp_blas_array_norm2, 114
fasp_blas_array_norminf, 114
blas_blc.c, 115
fasp_blas_bdbsr_aAxpy, 116
fasp_blas_bdbsr_mxv, 116
fasp_blas_dblc_aAxpy, 117
fasp_blas_dblc_mxv, 117
blas_bsr.c, 118
fasp_blas_dbsr_aAxpby, 119
fasp_blas_dbsr_aAxpy, 119
fasp_blas_dbsr_aAxpy_agg, 120
fasp_blas_dbsr_axm, 121
fasp_blas_dbsr_mxm, 121
fasp_blas_dbsr_mxv, 122
fasp_blas_dbsr_mxv_agg, 122
fasp_blas_dbsr_rap, 123
fasp_blas_dbsr_rap1, 124
fasp_blas_dbsr_rap_agg, 124
blas_csr.c, 125
fasp_blas_dcsr_aAxpy, 127
fasp_blas_dcsr_aAxpy_agg, 127
fasp_blas_dcsr_add, 128
fasp_blas_dcsr_axm, 128
fasp_blas_dcsr_bandwith, 129
fasp_blas_dcsr_mxm, 129
fasp_blas_dcsr_mxv, 130
fasp_blas_dcsr_mxv_agg, 131
fasp_blas_dcsr_ptap, 131
fasp_blas_dcsr_rap, 132
fasp_blas_dcsr_rap4, 133
fasp_blas_dcsr_rap_agg, 133
fasp_blas_dcsr_rap_agg1, 134
fasp_blas_dcsr_vmv, 135
blas_csrl.c, 135
fasp_blas_dcsrl_mxv, 136
blas_smat.c, 136
fasp_blas_array_axpy_nc2, 138
fasp_blas_array_axpy_nc3, 139
fasp_blas_array_axpy_nc5, 139
fasp_blas_array_axpy_nc7, 141
fasp_blas_array_axpyz_nc2, 141
fasp_blas_array_axpyz_nc3, 142
fasp_blas_array_axpyz_nc5, 143
fasp_blas_array_axpyz_nc7, 143
fasp_blas_smat_aAxpby, 144
fasp_blas_smat_add, 145
fasp_blas_smat_axm, 145
fasp_blas_smat_mul, 146
fasp_blas_smat_mul_nc2, 146
fasp_blas_smat_mul_nc3, 147
fasp_blas_smat_mul_nc5, 148
fasp_blas_smat_mul_nc7, 148
fasp_blas_smat_mxv, 149
fasp_blas_smat_mxv_nc2, 149
fasp_blas_smat_mxv_nc3, 150
fasp_blas_smat_mxv_nc5, 150
fasp_blas_smat_mxv_nc7, 152
fasp_blas_smat_ymAx, 152
fasp_blas_smat_ymAx_nc2, 153
fasp_blas_smat_ymAx_nc3, 154
fasp_blas_smat_ymAx_nc5, 154
fasp_blas_smat_ymAx_nc7, 155
fasp_blas_smat_ymAx_ns, 155
fasp_blas_smat_ymAx_ns2, 156
fasp_blas_smat_ymAx_ns3, 157
fasp_blas_smat_ymAx_ns5, 157
fasp_blas_smat_ymAx_ns7, 158

fasp_blas_smat_ypAx, 159
fasp_blas_smat_ypAx_nc2, 159
fasp_blas_smat_ypAx_nc3, 160
fasp_blas_smat_ypAx_nc5, 160
fasp_blas_smat_ypAx_nc7, 162
blas_str.c, 162
fasp_blas_dstr_aAxpy, 163
fasp_blas_dstr_mxv, 163
fasp_dstr_diagscale, 164
blas_vec.c, 165
fasp_blas_dvec_axpy, 165
fasp_blas_dvec_axpyz, 166
fasp_blas_dvec_dotprod, 166
fasp_blas_dvec_norm1, 167
fasp_blas_dvec_norm2, 168
fasp_blas_dvec_norminf, 168
fasp_blas_dvec_relerr, 169
block_BSR, 28
fasp_block.h, 196
block_Reservoir, 31
fasp_block.h, 196
block_dvector, 29
fasp_block.h, 196
block_iCSRmat, 30
fasp_block.h, 196
block_ivector, 30
fasp_block.h, 196

CF_ORDER
fasp_const.h, 201
CGPT
fasp_const.h, 201
CLASSIC_AMG
fasp_const.h, 201
COARSE_AC
fasp_const.h, 202
COARSE_CR
fasp_const.h, 202
COARSE_MIS
fasp_const.h, 202
COARSE_RSP
fasp_const.h, 202
COARSE_RS
fasp_const.h, 202
CPFIRST
fasp_const.h, 203
checkmat.c, 170
fasp_check_dCSRmat, 170
fasp_check_diagdom, 171
fasp_check_diagpos, 171
fasp_check_diagzero, 172
fasp_check_iCSRmat, 173
fasp_check_symm, 173
coarsening_cr.c, 174
fasp_amg_coarsening_cr, 174
coarsening_rs.c, 175
fasp_amg_coarsening_rs, 175
convert.c, 176
endian_convert_int, 177
endian_convert_real, 177
fasp_aux_bbyteToldouble, 178
fasp_aux_change_endian4, 178
fasp_aux_change_endian8, 179
count
fasp.h, 192

dBLCmat, 32
fasp_block.h, 196
dBSRmat, 32
fasp_block.h, 197
JA, 33
val, 33
dCOOmat, 33
fasp.h, 190
dCSRLmat, 34
fasp.h, 190
dCSRmat, 35
fasp.h, 190
dCSRmat2SAMGInput
interface_samg.c, 274
DESCEND
fasp_const.h, 203
DIAGONAL_PREF
fasp.h, 185
DLMALLOC
fasp.h, 186
dSTRmat, 37
fasp.h, 191
ddenmat, 36
fasp.h, 190
diag
precond_block_reservoir_data, 69
diaginv
precond_FASP_blkoil_data, 80
precond_block_reservoir_data, 69
diaginv_noscale
precond_FASP_blkoil_data, 80
diaginv_S
precond_FASP_blkoil_data, 80
diaginvS
precond_block_reservoir_data, 69
dlength
io.c, 308
doxygen.h, 180
dvector, 38
fasp.h, 191
dvector2SAMGInput
interface_samg.c, 274

e
grid2d, 39
ERROR_ALLOC_MEM
fasp_const.h, 203
ERROR_AMG_COARSE_TYPE
fasp_const.h, 203
ERROR_AMG_COARSEING
fasp_const.h, 203
ERROR_AMG_INTERP_TYPE
fasp_const.h, 203
ERROR_AMG_SMOOTH_TYPE
fasp_const.h, 204
ERROR_DATA_STRUCTURE
fasp_const.h, 204
ERROR_DATA_ZERODIAG
fasp_const.h, 204
ERROR_DUMMY_VAR
fasp_const.h, 204
ERROR_INPUT_PAR
fasp_const.h, 204
ERROR_LIC_TYPE
fasp_const.h, 204
ERROR_MAT_SIZE
fasp_const.h, 205
ERROR_MISC
fasp_const.h, 205
ERROR_NUM_BLOCKS
fasp_const.h, 205
ERROR_OPEN_FILE
fasp_const.h, 205
ERROR_QUAD_DIM
fasp_const.h, 205
ERROR_QUAD_TYPE
fasp_const.h, 205
ERROR_REGRESS
fasp_const.h, 206
ERROR_SOLVER_EXIT
fasp_const.h, 206
ERROR_SOLVER_ILUSETUP
fasp_const.h, 206
ERROR_SOLVER_MAXIT
fasp_const.h, 206
ERROR_SOLVER_MISC
fasp_const.h, 206
ERROR_SOLVER_PRECTYPE
fasp_const.h, 206
ERROR_SOLVER_SOLSTAG
fasp_const.h, 207
ERROR_SOLVER_STAG
fasp_const.h, 207
ERROR_SOLVER_TOLSMALL
fasp_const.h, 207
ERROR_SOLVER_TYPE
fasp_const.h, 207
ERROR_UNKNOWN
fasp_const.h, 207
ERROR_WRONG_FILE
fasp_const.h, 207
edges
grid2d, 39
ediri
grid2d, 39
efather
grid2d, 39
eigen.c, 180
fasp_dcsr_eig, 180
endian_convert_int
convert.c, 177
endian_convert_real
convert.c, 177

FALSE
fasp_const.h, 208
FASP_GET_START_END
threads.c, 583
FASP_GSRB
fasp.h, 186
FASP_SUCCESS
fasp_const.h, 208
FASP_USE_ILU
fasp.h, 186
FASP_VERSION
fasp.h, 186
FGPT
fasp_const.h, 208
FPFIRST
fasp_const.h, 208
famg.c, 181
fasp_solver_famg, 181
fasp.h, 182
__FASP_HEADER__, 185
ABS, 185
count, 192
dCOOmat, 190
dCSRLmat, 190
dCSRmat, 190
DIAGONAL_PREF, 185
DLMALLOC, 186
dSTRmat, 191
ddenmat, 190
dvector, 191
FASP_GSRB, 186
FASP_USE_ILU, 186
FASP_VERSION, 186
GE, 186
grid2d, 191
GT, 187
iCOOmat, 191

iCSRmat, 191
ILU_C_VERSION, 187
IMAP, 192
INT, 187
ISNAN, 187
idenmat, 191
ivector, 191
LONGLONG, 188
LONG, 188
LE, 187
LinkList, 192
ListElement, 192
LS, 188
MAXIMAP, 193
MAX, 188
MIN, 188
NEDMALLOC, 189
nx_rb, 193
ny_rb, 193
nz_rb, 193
PUT_INT, 189
PUT_REAL, 189
pcgrid2d, 192
pgrid2d, 192
REAL, 189
RS_C1, 189
SHORT, 190
total_alloc_count, 193
total_alloc_mem, 193
fasp_BinarySearch
ordering.c, 353
fasp_Schwarz_data_free
init.c, 267
fasp_Schwarz_get_block_matrix
schwarz_setup.c, 457
fasp_Schwarz_setup
schwarz_setup.c, 458
fasp_amg_amli_coef
amlirecur.c, 100
fasp_amg_coarsening_cr
coarsening_cr.c, 174
fasp_amg_coarsening_rs
coarsening_rs.c, 175
fasp_amg_data_bsr_create
init.c, 262
fasp_amg_data_bsr_free
init.c, 263
fasp_amg_data_create
init.c, 263
fasp_amg_data_free
init.c, 264
fasp_amg_interp
interpolation.c, 278
fasp_amg_interp1
interpolation.c, 278
fasp_amg_interp_em
interpolation_em.c, 280
fasp_amg_interp_trunc
interpolation.c, 279
fasp_amg_setup_cr
amg_setup_cr.c, 91
fasp_amg_setup_rs
amg_setup_rs.c, 92
fasp_amg_setup_sa
amg_setup_sa.c, 93
fasp_amg_setup_sa_bsr
amg_setup_sa.c, 94
fasp_amg_setup_ua
amg_setup_ua.c, 95
fasp_amg_setup_ua_bsr
amg_setup_ua.c, 96
fasp_amg_solve
amg_solve.c, 97
fasp_amg_solve_amli
amg_solve.c, 98
fasp_amg_solve_nl_amli
amg_solve.c, 98
fasp_array_cp
array.c, 104
fasp_array_cp_nc3
array.c, 104
fasp_array_cp_nc5
array.c, 105
fasp_array_cp_nc7
array.c, 105
fasp_array_invpermut_nb
array.c, 106
fasp_array_null
array.c, 106
fasp_array_permut_nb
array.c, 107
fasp_array_set
array.c, 108
fasp_aux_bbyteToldouble
convert.c, 178
fasp_aux_change_endian4
convert.c, 178
fasp_aux_change_endian8
convert.c, 179
fasp_aux_dQuickSort
ordering.c, 348
fasp_aux_dQuickSortIndex
ordering.c, 349
fasp_aux_givens
givens.c, 233
fasp_aux_iQuickSort
ordering.c, 350
fasp_aux_iQuickSortIndex

ordering.c, 350
fasp_aux_merge
ordering.c, 351
fasp_aux_msort
ordering.c, 352
fasp_aux_unique
ordering.c, 352
fasp_blas_array_ax
blas_array.c, 110
fasp_blas_array_axpby
blas_array.c, 111
fasp_blas_array_axpy
blas_array.c, 111
fasp_blas_array_axpy_nc2
blas_smat.c, 138
fasp_blas_array_axpy_nc3
blas_smat.c, 139
fasp_blas_array_axpy_nc5
blas_smat.c, 139
fasp_blas_array_axpy_nc7
blas_smat.c, 141
fasp_blas_array_axpyz
blas_array.c, 112
fasp_blas_array_axpyz_nc2
blas_smat.c, 141
fasp_blas_array_axpyz_nc3
blas_smat.c, 142
fasp_blas_array_axpyz_nc5
blas_smat.c, 143
fasp_blas_array_axpyz_nc7
blas_smat.c, 143
fasp_blas_array_dotprod
blas_array.c, 113
fasp_blas_array_norm1
blas_array.c, 113
fasp_blas_array_norm2
blas_array.c, 114
fasp_blas_array_norminf
blas_array.c, 114
fasp_blas_bdbsr_aAxpy
blas_blc.c, 116
fasp_blas_bdbsr_mxv
blas_blc.c, 116
fasp_blas_dblc_aAxpy
blas_blc.c, 117
fasp_blas_dblc_mxv
blas_blc.c, 117
fasp_blas_dbsr_aAxpby
blas_bsr.c, 119
fasp_blas_dbsr_aAxpy
blas_bsr.c, 119
fasp_blas_dbsr_aAxpy_agg
blas_bsr.c, 120
fasp_blas_dbsr_axm
blas_bsr.c, 121
fasp_blas_dbsr_mxm
blas_bsr.c, 121
fasp_blas_dbsr_mxv
blas_bsr.c, 122
fasp_blas_dbsr_mxv_agg
blas_bsr.c, 122
fasp_blas_dbsr_rap
blas_bsr.c, 123
fasp_blas_dbsr_rap1
blas_bsr.c, 124
fasp_blas_dbsr_rap_agg
blas_bsr.c, 124
fasp_blas_dcsr_aAxpy
blas_csr.c, 127
fasp_blas_dcsr_aAxpy_agg
blas_csr.c, 127
fasp_blas_dcsr_add
blas_csr.c, 128
fasp_blas_dcsr_axm
blas_csr.c, 128
fasp_blas_dcsr_bandwith
blas_csr.c, 129
fasp_blas_dcsr_mxm
blas_csr.c, 129
fasp_blas_dcsr_mxv
blas_csr.c, 130
fasp_blas_dcsr_mxv_agg
blas_csr.c, 131
fasp_blas_dcsr_ptap
blas_csr.c, 131
fasp_blas_dcsr_rap
blas_csr.c, 132
fasp_blas_dcsr_rap2
rap.c, 455
fasp_blas_dcsr_rap4
blas_csr.c, 133
fasp_blas_dcsr_rap_agg
blas_csr.c, 133
fasp_blas_dcsr_rap_agg1
blas_csr.c, 134
fasp_blas_dcsr_vmv
blas_csr.c, 135
fasp_blas_dcsrl_mxv
blas_csrl.c, 136
fasp_blas_dstr_aAxpy
blas_str.c, 163
fasp_blas_dstr_mxv
blas_str.c, 163
fasp_blas_dvec_axpy
blas_vec.c, 165
fasp_blas_dvec_axpyz
blas_vec.c, 166
fasp_blas_dvec_dotprod

blas_vec.c, 166
fasp_blas_dvec_norm1
blas_vec.c, 167
fasp_blas_dvec_norm2
blas_vec.c, 168
fasp_blas_dvec_norminf
blas_vec.c, 168
fasp_blas_dvec_relerr
blas_vec.c, 169
fasp_blas_smat_Linfinity
smat.c, 464
fasp_blas_smat_aAxpby
blas_smat.c, 144
fasp_blas_smat_add
blas_smat.c, 145
fasp_blas_smat_axm
blas_smat.c, 145
fasp_blas_smat_inv
smat.c, 460
fasp_blas_smat_inv_nc
smat.c, 461
fasp_blas_smat_inv_nc2
smat.c, 461
fasp_blas_smat_inv_nc3
smat.c, 462
fasp_blas_smat_inv_nc4
smat.c, 462
fasp_blas_smat_inv_nc5
smat.c, 463
fasp_blas_smat_inv_nc7
smat.c, 463
fasp_blas_smat_invp_nc
smat.c, 464
fasp_blas_smat_mul
blas_smat.c, 146
fasp_blas_smat_mul_nc2
blas_smat.c, 146
fasp_blas_smat_mul_nc3
blas_smat.c, 147
fasp_blas_smat_mul_nc5
blas_smat.c, 148
fasp_blas_smat_mul_nc7
blas_smat.c, 148
fasp_blas_smat_mxv
blas_smat.c, 149
fasp_blas_smat_mxv_nc2
blas_smat.c, 149
fasp_blas_smat_mxv_nc3
blas_smat.c, 150
fasp_blas_smat_mxv_nc5
blas_smat.c, 150
fasp_blas_smat_mxv_nc7
blas_smat.c, 152
fasp_blas_smat_ymAx
blas_smat.c, 152
fasp_blas_smat_ymAx_nc2
blas_smat.c, 153
fasp_blas_smat_ymAx_nc3
blas_smat.c, 154
fasp_blas_smat_ymAx_nc5
blas_smat.c, 154
fasp_blas_smat_ymAx_nc7
blas_smat.c, 155
fasp_blas_smat_ymAx_ns
blas_smat.c, 155
fasp_blas_smat_ymAx_ns2
blas_smat.c, 156
fasp_blas_smat_ymAx_ns3
blas_smat.c, 157
fasp_blas_smat_ymAx_ns5
blas_smat.c, 157
fasp_blas_smat_ymAx_ns7
blas_smat.c, 158
fasp_blas_smat_ypAx
blas_smat.c, 159
fasp_blas_smat_ypAx_nc2
blas_smat.c, 159
fasp_blas_smat_ypAx_nc3
blas_smat.c, 160
fasp_blas_smat_ypAx_nc5
blas_smat.c, 160
fasp_blas_smat_ypAx_nc7
blas_smat.c, 162
fasp_block.h, 194
__FASPBLOCK_HEADER__, 195
block_BSR, 196
block_Reservoir, 196
block_dvector, 196
block_iCSRmat, 196
block_ivector, 196
dBLCmat, 196
dBSRmat, 197
precond_block_reservoir_data, 197
SMOOTHER_BLKOIL, 195
SMOOTHER_SPETEN, 195
fasp_check_dCSRmat
checkmat.c, 170
fasp_check_diagdom
checkmat.c, 171
fasp_check_diagpos
checkmat.c, 171
fasp_check_diagzero
checkmat.c, 172
fasp_check_iCSRmat
checkmat.c, 173
fasp_check_symm
checkmat.c, 173
fasp_chkerr

message.c, 342
fasp_const.h, 197
AMLI_CYCLE, 201
ASCEND, 201
BIGREAL, 201
CF_ORDER, 201
CGPT, 201
CLASSIC_AMG, 201
COARSE_AC, 202
COARSE_CR, 202
COARSE_MIS, 202
COARSE_RSP, 202
COARSE_RS, 202
CPFIRST, 203
DESCEND, 203
ERROR_ALLOC_MEM, 203
ERROR_AMG_COARSE_TYPE, 203
ERROR_AMG_COARSEING, 203
ERROR_AMG_INTERP_TYPE, 203
ERROR_AMG_SMOOTH_TYPE, 204
ERROR_DATA_STRUCTURE, 204
ERROR_DATA_ZERODIAG, 204
ERROR_DUMMY_VAR, 204
ERROR_INPUT_PAR, 204
ERROR_LIC_TYPE, 204
ERROR_MAT_SIZE, 205
ERROR_MISC, 205
ERROR_NUM_BLOCKS, 205
ERROR_OPEN_FILE, 205
ERROR_QUAD_DIM, 205
ERROR_QUAD_TYPE, 205
ERROR_REGRESS, 206
ERROR_SOLVER_EXIT, 206
ERROR_SOLVER_ILUSETUP, 206
ERROR_SOLVER_MAXIT, 206
ERROR_SOLVER_MISC, 206
ERROR_SOLVER_PRECTYPE, 206
ERROR_SOLVER_SOLSTAG, 207
ERROR_SOLVER_STAG, 207
ERROR_SOLVER_TOLSMALL, 207
ERROR_SOLVER_TYPE, 207
ERROR_UNKNOWN, 207
ERROR_WRONG_FILE, 207
FALSE, 208
FASP_SUCCESS, 208
FGPT, 208
FPFIRST, 208
G0PT, 208
ILU_MC_OMP, 209
ILUk, 209
ILUt, 209
ILUtp, 209
INTERP_DIR, 209
INTERP_ENG, 210
INTERP_STD, 210
ISPT, 210
MAT_BLC, 210
MAT_BSR, 211
MAT_CSRL, 211
MAT_CSR, 211
MAT_FREE, 211
MAT_STR, 211
MAT_SymCSR, 212
MAT_bBSR, 210
MAX_AMG_LVL, 212
MAX_CRATE, 212
MAX_REFINE_LVL, 212
MAX_RESTART, 212
MAX_STAG, 212
MIN_CDOF, 213
MIN_CRATE, 213
NL_AMLI_CYCLE, 213
NO_ORDER, 213
OFF, 213
OPENMP_HOLDS, 214
ON, 214
PAIRWISE, 214
PREC_AMG, 214
PREC_DIAG, 214
PREC_FMG, 215
PREC_ILU, 215
PREC_NULL, 215
PREC_SCHWARZ, 215
PRINT_ALL, 215
PRINT_MIN, 216
PRINT_MORE, 216
PRINT_MOST, 216
PRINT_NONE, 216
PRINT_SOME, 216
SA_AMG, 217
SCHWARZ_BACKWARD, 217
SCHWARZ_FORWARD, 217
SCHWARZ_SYMMETRIC, 217
SMALLREAL2, 218
SMALLREAL, 217
SMOOTHER_CG, 218
SMOOTHER_GSOR, 218
SMOOTHER_GS, 218
SMOOTHER_JACOBI, 218
SMOOTHER_L1DIAG, 218
SMOOTHER_POLY, 219
SMOOTHER_SGSOR, 219
SMOOTHER_SGS, 219
SMOOTHER_SOR, 219
SMOOTHER_SSOR, 219
SOLVER_AMG, 219
SOLVER_BiCGstab, 220
SOLVER_CG, 220

SOLVER_DEFAULT, 220
SOLVER_FMG, 220
SOLVER_GCG, 220
SOLVER_GCR, 221
SOLVER_GMRES, 221
SOLVER_MUMPS, 221
SOLVER_MinRes, 221
SOLVER_PARDISO, 221
SOLVER_SBiCGstab, 221
SOLVER_SCG, 222
SOLVER_SGCG, 222
SOLVER_SGMRES, 222
SOLVER_SMinRes, 222
SOLVER_SUPERLU, 222
SOLVER_SVFGMRES, 222
SOLVER_SVGMRES, 223
SOLVER_UMFPACK, 223
SOLVER_VBiCGstab, 223
SOLVER_VFGMRES, 223
SOLVER_VGMRES, 223
STAG_RATIO, 223
STOP_MOD_REL_RES, 224
STOP_REL_PRECRES, 224
STOP_REL_RES, 224
TRUE, 224
UA_AMG, 224
UNPT, 225
USERDEFINED, 225
V_CYCLE, 225
VMB, 225
W_CYCLE, 225
fasp_dblc_free
sparse_block.c, 507
fasp_dbsr_Linfinity_dcsr
sparse_block.c, 509
fasp_dbsr_alloc
sparse_bsr.c, 511
fasp_dbsr_cp
sparse_bsr.c, 512
fasp_dbsr_create
sparse_bsr.c, 512
fasp_dbsr_diagLU2
sparse_bsr.c, 517
fasp_dbsr_diagLU
sparse_bsr.c, 517
fasp_dbsr_diaginv
sparse_bsr.c, 513
fasp_dbsr_diaginv2
sparse_bsr.c, 514
fasp_dbsr_diaginv3
sparse_bsr.c, 514
fasp_dbsr_diaginv4
sparse_bsr.c, 516
fasp_dbsr_diagpref
sparse_bsr.c, 518
fasp_dbsr_free
sparse_bsr.c, 519
fasp_dbsr_getblk
sparse_block.c, 507
fasp_dbsr_getblk_dcsr
sparse_block.c, 508
fasp_dbsr_getdiag
sparse_bsr.c, 519
fasp_dbsr_getdiaginv
sparse_bsr.c, 520
fasp_dbsr_null
sparse_bsr.c, 520
fasp_dbsr_perm
sparse_bsr.c, 521
fasp_dbsr_plot
graphics.c, 243
fasp_dbsr_print
io.c, 283
fasp_dbsr_read
io.c, 284
fasp_dbsr_subplot
graphics.c, 244
fasp_dbsr_trans
sparse_bsr.c, 522
fasp_dbsr_write
io.c, 284
fasp_dbsr_write_coo
io.c, 286
fasp_dcoo1_read
io.c, 286
fasp_dcoo_alloc
sparse_coo.c, 523
fasp_dcoo_create
sparse_coo.c, 523
fasp_dcoo_free
sparse_coo.c, 524
fasp_dcoo_print
io.c, 287
fasp_dcoo_read
io.c, 288
fasp_dcoo_shift
sparse_coo.c, 525
fasp_dcoo_shift_read
io.c, 288
fasp_dcoo_write
io.c, 289
fasp_dcsr_CMK_order
ordering.c, 354
fasp_dcsr_RCMK_order
ordering.c, 354
fasp_dcsr_Schwarz_backward_smoother
schwarz_setup.c, 456
fasp_dcsr_Schwarz_forward_smoother

schwarz_setup.c, 457
fasp_dcsr_alloc
sparse_csr.c, 527
fasp_dcsr_compress
sparse_csr.c, 527
fasp_dcsr_compress_inplace
sparse_csr.c, 528
fasp_dcsr_cp
sparse_csr.c, 529
fasp_dcsr_create
sparse_csr.c, 529
fasp_dcsr_diagpref
sparse_csr.c, 530
fasp_dcsr_eig
eigen.c, 180
fasp_dcsr_free
sparse_csr.c, 531
fasp_dcsr_getblk
sparse_block.c, 509
fasp_dcsr_getcol
sparse_csr.c, 531
fasp_dcsr_getdiag
sparse_csr.c, 532
fasp_dcsr_multicoloring
sparse_csr.c, 532
fasp_dcsr_null
sparse_csr.c, 533
fasp_dcsr_perm
sparse_csr.c, 533
fasp_dcsr_permz
sparse_csr.c, 534
fasp_dcsr_plot
graphics.c, 245
fasp_dcsr_print
io.c, 290
fasp_dcsr_read
io.c, 290
fasp_dcsr_regdiag
sparse_csr.c, 535
fasp_dcsr_shift
sparse_csr.c, 535
fasp_dcsr_sort
sparse_csr.c, 536
fasp_dcsr_sortz
sparse_csr.c, 536
fasp_dcsr_subplot
graphics.c, 245
fasp_dcsr_symdiagscale
sparse_csr.c, 537
fasp_dcsr_sympat
sparse_csr.c, 537
fasp_dcsr_trans
sparse_csr.c, 538
fasp_dcsr_transz
sparse_csr.c, 539
fasp_dcsr_write_coo
io.c, 291
fasp_dcsrl_create
sparse_csrl.c, 544
fasp_dcsrl_free
sparse_csrl.c, 545
fasp_dcsrvec1_read
io.c, 291
fasp_dcsrvec1_write
io.c, 292
fasp_dcsrvec2_read
io.c, 293
fasp_dcsrvec2_write
io.c, 294
fasp_dmtx_read
io.c, 295
fasp_dmtxsym_read
io.c, 295
fasp_dstr_alloc
sparse_str.c, 546
fasp_dstr_cp
sparse_str.c, 547
fasp_dstr_create
sparse_str.c, 547
fasp_dstr_diagscale
blas_str.c, 164
fasp_dstr_free
sparse_str.c, 548
fasp_dstr_null
sparse_str.c, 548
fasp_dstr_print
io.c, 296
fasp_dstr_read
io.c, 296
fasp_dstr_write
io.c, 297
fasp_dvec_alloc
vec.c, 587
fasp_dvec_cp
vec.c, 587
fasp_dvec_create
vec.c, 587
fasp_dvec_free
vec.c, 588
fasp_dvec_isnan
vec.c, 588
fasp_dvec_maxdiff
vec.c, 589
fasp_dvec_null
vec.c, 590
fasp_dvec_print
io.c, 298
fasp_dvec_rand

vec.c, 590
fasp_dvec_read
io.c, 298
fasp_dvec_set
vec.c, 591
fasp_dvec_symdiagscale
vec.c, 592
fasp_dvec_write
io.c, 299
fasp_dvecind_read
io.c, 299
fasp_dvecind_write
io.c, 300
fasp_famg_solve
amg_solve.c, 99
fasp_format_dblc_dcsr
formats.c, 228
fasp_format_dbsr_dcoo
formats.c, 228
fasp_format_dbsr_dcsr
formats.c, 229
fasp_format_dcoo_dcsr
formats.c, 229
fasp_format_dcsr_dbsr
formats.c, 230
fasp_format_dcsr_dcoo
formats.c, 230
fasp_format_dcsrl_dcsr
formats.c, 231
fasp_format_dstr_dbsr
formats.c, 232
fasp_format_dstr_dcsr
formats.c, 232
fasp_fwrapper_amg_
wrapper.c, 595
fasp_fwrapper_krylov_amg_
wrapper.c, 596
fasp_gauss2d
quadrature.c, 453
fasp_generate_diaginv_block
smoother_str.c, 494
fasp_gettime
timing.c, 585
fasp_grid2d_plot
graphics.c, 246
fasp_hb_read
io.c, 301
fasp_iarray_cp
array.c, 108
fasp_iarray_set
array.c, 109
fasp_icsr_cp
sparse_csr.c, 539
fasp_icsr_create
sparse_csr.c, 540
fasp_icsr_free
sparse_csr.c, 540
fasp_icsr_null
sparse_csr.c, 542
fasp_icsr_trans
sparse_csr.c, 542
fasp_iden_free
smat.c, 465
fasp_ilu_data_alloc
init.c, 264
fasp_ilu_data_free
init.c, 265
fasp_ilu_data_null
init.c, 265
fasp_ilu_dbsr_setup
ilu_setup_bsr.c, 256
fasp_ilu_dbsr_setup_levsch_omp
ilu_setup_bsr.c, 256
fasp_ilu_dbsr_setup_mc_omp
ilu_setup_bsr.c, 257
fasp_ilu_dbsr_setup_omp
ilu_setup_bsr.c, 258
fasp_ilu_dcsr_setup
ilu_setup_csr.c, 259
fasp_ilu_dstr_setup0
ilu_setup_str.c, 260
fasp_ilu_dstr_setup1
ilu_setup_str.c, 261
fasp_iluk
ilu.c, 248
fasp_ilut
ilu.c, 248
fasp_ilutp
ilu.c, 249
fasp_ivec_alloc
vec.c, 592
fasp_ivec_create
vec.c, 593
fasp_ivec_free
vec.c, 593
fasp_ivec_print
io.c, 301
fasp_ivec_read
io.c, 302
fasp_ivec_set
vec.c, 594
fasp_ivec_write
io.c, 303
fasp_ivecind_read
io.c, 303
fasp_matrix_read
io.c, 304
fasp_matrix_read_bin

io.c, 305
fasp_matrix_write
io.c, 305
fasp_mem_calloc
memory.c, 337
fasp_mem_check
memory.c, 337
fasp_mem_dcsr_check
memory.c, 338
fasp_mem_free
memory.c, 338
fasp_mem_iludata_check
memory.c, 339
fasp_mem_realloc
memory.c, 340
fasp_mem_usage
memory.c, 340
fasp_multicolors_independent_set
ordering.c, 355
fasp_param_Schwarz_init
parameters.c, 364
fasp_param_Schwarz_print
parameters.c, 364
fasp_param_Schwarz_set
parameters.c, 365
fasp_param_amg_init
parameters.c, 357
fasp_param_amg_print
parameters.c, 357
fasp_param_amg_set
parameters.c, 359
fasp_param_amg_to_prec
parameters.c, 359
fasp_param_amg_to_prec_bsr
parameters.c, 360
fasp_param_check
input.c, 268
fasp_param_ilu_init
parameters.c, 360
fasp_param_ilu_print
parameters.c, 361
fasp_param_ilu_set
parameters.c, 361
fasp_param_init
parameters.c, 362
fasp_param_input
input.c, 268
fasp_param_input_init
parameters.c, 362
fasp_param_prec_to_amg
parameters.c, 363
fasp_param_prec_to_amg_bsr
parameters.c, 363
fasp_param_set
parameters.c, 365
fasp_param_solver_init
parameters.c, 366
fasp_param_solver_print
parameters.c, 366
fasp_param_solver_set
parameters.c, 367
fasp_poisson_fgmg_1D
gmg_poisson.c, 235
fasp_poisson_fgmg_2D
gmg_poisson.c, 236
fasp_poisson_fgmg_3D
gmg_poisson.c, 236
fasp_poisson_gmg_1D
gmg_poisson.c, 238
fasp_poisson_gmg_2D
gmg_poisson.c, 239
fasp_poisson_gmg_3D
gmg_poisson.c, 239
fasp_poisson_pcg_gmg_1D
gmg_poisson.c, 240
fasp_poisson_pcg_gmg_2D
gmg_poisson.c, 241
fasp_poisson_pcg_gmg_3D
gmg_poisson.c, 242
fasp_precond_Schwarz
precond_csr.c, 434
fasp_precond_amg
precond_csr.c, 426
fasp_precond_amg_nk
precond_csr.c, 427
fasp_precond_amli
precond_csr.c, 427
fasp_precond_block_SGS_3
precond_blc.c, 412
fasp_precond_block_SGS_3_amg
precond_blc.c, 412
fasp_precond_block_diag_3
precond_blc.c, 407
fasp_precond_block_diag_3_amg
precond_blc.c, 407
fasp_precond_block_diag_4
precond_blc.c, 408
fasp_precond_block_lower_3
precond_blc.c, 408
fasp_precond_block_lower_3_amg
precond_blc.c, 410
fasp_precond_block_lower_4
precond_blc.c, 410
fasp_precond_block_upper_3
precond_blc.c, 414
fasp_precond_block_upper_3_amg
precond_blc.c, 414
fasp_precond_data_null

init.c, 266
fasp_precond_dbsr_amg
precond_bsr.c, 417
fasp_precond_dbsr_amg_nk
precond_bsr.c, 418
fasp_precond_dbsr_diag
precond_bsr.c, 418
fasp_precond_dbsr_diag_nc2
precond_bsr.c, 420
fasp_precond_dbsr_diag_nc3
precond_bsr.c, 421
fasp_precond_dbsr_diag_nc5
precond_bsr.c, 421
fasp_precond_dbsr_diag_nc7
precond_bsr.c, 422
fasp_precond_dbsr_ilu
precond_bsr.c, 423
fasp_precond_dbsr_ilu_levsch_omp
precond_bsr.c, 424
fasp_precond_dbsr_ilu_mc_omp
precond_bsr.c, 424
fasp_precond_dbsr_nl_amli
precond_bsr.c, 425
fasp_precond_diag
precond_csr.c, 429
fasp_precond_dstr_blockgs
precond_str.c, 436
fasp_precond_dstr_diag
precond_str.c, 436
fasp_precond_dstr_ilu0
precond_str.c, 437
fasp_precond_dstr_ilu0_backward
precond_str.c, 437
fasp_precond_dstr_ilu0_forward
precond_str.c, 438
fasp_precond_dstr_ilu1
precond_str.c, 438
fasp_precond_dstr_ilu1_backward
precond_str.c, 440
fasp_precond_dstr_ilu1_forward
precond_str.c, 440
fasp_precond_famg
precond_csr.c, 429
fasp_precond_free
precond_csr.c, 431
fasp_precond_ilu
precond_csr.c, 431
fasp_precond_ilu_backward
precond_csr.c, 432
fasp_precond_ilu_forward
precond_csr.c, 433
fasp_precond_nl_amli
precond_csr.c, 433
fasp_precond_null
init.c, 266
fasp_precond_setup
precond_csr.c, 434
fasp_precond_sweeping
precond_blc.c, 416
fasp_qsplit
ilu.c, 251
fasp_quad2d
quadrature.c, 454
fasp_set_GS_threads
threads.c, 584
fasp_smat_identity
smat.c, 465
fasp_smat_identity_nc2
smat.c, 466
fasp_smat_identity_nc3
smat.c, 466
fasp_smat_identity_nc5
smat.c, 467
fasp_smat_identity_nc7
smat.c, 467
fasp_smat_lu_decomp
lu.c, 334
fasp_smat_lu_solve
lu.c, 335
fasp_smoother_dbsr_gs
smoother_bsr.c, 469
fasp_smoother_dbsr_gs1
smoother_bsr.c, 470
fasp_smoother_dbsr_gs_ascend
smoother_bsr.c, 470
fasp_smoother_dbsr_gs_ascend1
smoother_bsr.c, 471
fasp_smoother_dbsr_gs_descend
smoother_bsr.c, 472
fasp_smoother_dbsr_gs_descend1
smoother_bsr.c, 472
fasp_smoother_dbsr_gs_order1
smoother_bsr.c, 473
fasp_smoother_dbsr_gs_order2
smoother_bsr.c, 474
fasp_smoother_dbsr_ilu
smoother_bsr.c, 474
fasp_smoother_dbsr_jacobi
smoother_bsr.c, 475
fasp_smoother_dbsr_jacobi1
smoother_bsr.c, 475
fasp_smoother_dbsr_jacobi_setup
smoother_bsr.c, 476
fasp_smoother_dbsr_sor
smoother_bsr.c, 477
fasp_smoother_dbsr_sor1
smoother_bsr.c, 477
fasp_smoother_dbsr_sor_ascend

smoother_bsr.c, 478
fasp_smoother_dbsr_sor_descend
smoother_bsr.c, 479
fasp_smoother_dbsr_sor_order
smoother_bsr.c, 479
fasp_smoother_dcsr_L1diag
smoother_csr.c, 486
fasp_smoother_dcsr_gs
smoother_csr.c, 482
fasp_smoother_dcsr_gs_cf
smoother_csr.c, 482
fasp_smoother_dcsr_gs_rb3d
smoother_csr.c, 483
fasp_smoother_dcsr_gscr
smoother_csr_cr.c, 489
fasp_smoother_dcsr_ilu
smoother_csr.c, 484
fasp_smoother_dcsr_jacobi
smoother_csr.c, 484
fasp_smoother_dcsr_kaczmarz
smoother_csr.c, 485
fasp_smoother_dcsr_poly
smoother_csr_poly.c, 491
fasp_smoother_dcsr_poly_old
smoother_csr_poly.c, 491
fasp_smoother_dcsr_sgs
smoother_csr.c, 486
fasp_smoother_dcsr_sor
smoother_csr.c, 487
fasp_smoother_dcsr_sor_cf
smoother_csr.c, 488
fasp_smoother_dstr_gs
smoother_str.c, 495
fasp_smoother_dstr_gs1
smoother_str.c, 495
fasp_smoother_dstr_gs_ascend
smoother_str.c, 496
fasp_smoother_dstr_gs_cf
smoother_str.c, 497
fasp_smoother_dstr_gs_descend
smoother_str.c, 497
fasp_smoother_dstr_gs_order
smoother_str.c, 499
fasp_smoother_dstr_jacobi
smoother_str.c, 499
fasp_smoother_dstr_jacobi1
smoother_str.c, 500
fasp_smoother_dstr_schwarz
smoother_str.c, 501
fasp_smoother_dstr_sor
smoother_str.c, 501
fasp_smoother_dstr_sor1
smoother_str.c, 502
fasp_smoother_dstr_sor_ascend
smoother_str.c, 503
fasp_smoother_dstr_sor_cf
smoother_str.c, 503
fasp_smoother_dstr_sor_descend
smoother_str.c, 505
fasp_smoother_dstr_sor_order
smoother_str.c, 506
fasp_solver_amg
amg.c, 89
fasp_solver_amli
amlirecur.c, 101
fasp_solver_dblc_itsolver
itsolver_blc.c, 309
fasp_solver_dblc_krylov
itsolver_blc.c, 310
fasp_solver_dblc_krylov_block_3
itsolver_blc.c, 311
fasp_solver_dblc_krylov_block_4
itsolver_blc.c, 311
fasp_solver_dblc_krylov_sweeping
itsolver_blc.c, 312
fasp_solver_dblc_pbcgs
pbcgs.c, 370
fasp_solver_dblc_pcg
pcg.c, 381
fasp_solver_dblc_pgmres
pgmres.c, 394
fasp_solver_dblc_pminres
pminres.c, 401
fasp_solver_dblc_pvbcgs
pbcgs.c, 370
fasp_solver_dblc_pvfgmres
pvfgmres.c, 443
fasp_solver_dblc_pvgmres
pvgmres.c, 448
fasp_solver_dblc_spbcgs
spbcgs.c, 561
fasp_solver_dblc_spcg
spcg.c, 566
fasp_solver_dblc_spgmres
spgmres.c, 570
fasp_solver_dblc_spminres
spminres.c, 575
fasp_solver_dblc_spvgmres
spvgmres.c, 578
fasp_solver_dbsr_itsolver
itsolver_bsr.c, 314
fasp_solver_dbsr_krylov
itsolver_bsr.c, 314
fasp_solver_dbsr_krylov_amg
itsolver_bsr.c, 315
fasp_solver_dbsr_krylov_amg_nk
itsolver_bsr.c, 316
fasp_solver_dbsr_krylov_diag

itsolver_bsr.c, 317
fasp_solver_dbsr_krylov_ilu
itsolver_bsr.c, 317
fasp_solver_dbsr_krylov_nk_amg
itsolver_bsr.c, 319
fasp_solver_dbsr_pbcgs
pbcgs.c, 371
fasp_solver_dbsr_pcg
pcg.c, 382
fasp_solver_dbsr_pgmres
pgmres.c, 395
fasp_solver_dbsr_pvbcgs
pbcgs.c, 372
fasp_solver_dbsr_pvfgmres
pvfgmres.c, 444
fasp_solver_dbsr_pvgmres
pvgmres.c, 449
fasp_solver_dbsr_spbcgs
spbcgs.c, 562
fasp_solver_dbsr_spgmres
spgmres.c, 570
fasp_solver_dbsr_spvgmres
spvgmres.c, 580
fasp_solver_dcsr_itsolver
itsolver_csr.c, 321
fasp_solver_dcsr_krylov
itsolver_csr.c, 321
fasp_solver_dcsr_krylov_Schwarz
itsolver_csr.c, 326
fasp_solver_dcsr_krylov_amg
itsolver_csr.c, 322
fasp_solver_dcsr_krylov_amg_nk
itsolver_csr.c, 323
fasp_solver_dcsr_krylov_diag
itsolver_csr.c, 324
fasp_solver_dcsr_krylov_ilu
itsolver_csr.c, 324
fasp_solver_dcsr_krylov_ilu_M
itsolver_csr.c, 325
fasp_solver_dcsr_pbcgs
pbcgs.c, 373
fasp_solver_dcsr_pcg
pcg.c, 383
fasp_solver_dcsr_pgcg
pgcg.c, 388
fasp_solver_dcsr_pgcr
pgcr.c, 391
fasp_solver_dcsr_pgcr1
pgcr.c, 392
fasp_solver_dcsr_pgmres
pgmres.c, 396
fasp_solver_dcsr_pminres
pminres.c, 401
fasp_solver_dcsr_pvbcgs
pbcgs.c, 374
fasp_solver_dcsr_pvfgmres
pvfgmres.c, 445
fasp_solver_dcsr_pvgmres
pvgmres.c, 449
fasp_solver_dcsr_spbcgs
spbcgs.c, 563
fasp_solver_dcsr_spcg
spcg.c, 567
fasp_solver_dcsr_spgmres
spgmres.c, 571
fasp_solver_dcsr_spminres
spminres.c, 576
fasp_solver_dcsr_spvgmres
spvgmres.c, 581
fasp_solver_dstr_itsolver
itsolver_str.c, 330
fasp_solver_dstr_krylov
itsolver_str.c, 331
fasp_solver_dstr_krylov_blockgs
itsolver_str.c, 331
fasp_solver_dstr_krylov_diag
itsolver_str.c, 332
fasp_solver_dstr_krylov_ilu
itsolver_str.c, 333
fasp_solver_dstr_pbcgs
pbcgs.c, 374
fasp_solver_dstr_pcg
pcg.c, 384
fasp_solver_dstr_pgmres
pgmres.c, 397
fasp_solver_dstr_pminres
pminres.c, 402
fasp_solver_dstr_pvbcgs
pbcgs.c, 375
fasp_solver_dstr_pvgmres
pvgmres.c, 450
fasp_solver_dstr_spbcgs
spbcgs.c, 564
fasp_solver_dstr_spcg
spcg.c, 568
fasp_solver_dstr_spgmres
spgmres.c, 572
fasp_solver_dstr_spminres
spminres.c, 577
fasp_solver_dstr_spvgmres
spvgmres.c, 582
fasp_solver_famg
famg.c, 181
fasp_solver_fmgcycle
fmgcycle.c, 226
fasp_solver_itsolver
itsolver_mf.c, 327
fasp_solver_itsolver_init

itsolver_mf.c, 328
fasp_solver_krylov
itsolver_mf.c, 329
fasp_solver_mgcycle
mgcycle.c, 345
fasp_solver_mgcycle_bsr
mgcycle.c, 346
fasp_solver_mgrecur
mgrecur.c, 347
fasp_solver_mumps
interface_mumps.c, 270
fasp_solver_mumps_steps
interface_mumps.c, 270
fasp_solver_nl_amli
amlirecur.c, 101
fasp_solver_nl_amli_bsr
amlirecur.c, 102
fasp_solver_pardiso
interface_pardiso.c, 272
fasp_solver_pbcgs
pbcgs_mf.c, 378
fasp_solver_pcg
pcg_mf.c, 386
fasp_solver_pgcg
pgcg_mf.c, 389
fasp_solver_pgmres
pgmres_mf.c, 398
fasp_solver_pminres
pminres_mf.c, 405
fasp_solver_pvbcgs
pbcgs_mf.c, 379
fasp_solver_pvfgmres
pvfgmres_mf.c, 446
fasp_solver_pvgmres
pvgmres_mf.c, 452
fasp_solver_superlu
interface_superlu.c, 275
fasp_solver_umfpack
interface_umfpack.c, 276
fasp_sparse_MIS
sparse_util.c, 554
fasp_sparse_aat_
sparse_util.c, 550
fasp_sparse_abyb_
sparse_util.c, 551
fasp_sparse_abybms_
sparse_util.c, 552
fasp_sparse_aplbms_
sparse_util.c, 552
fasp_sparse_aplusb_
sparse_util.c, 553
fasp_sparse_iit_
sparse_util.c, 554
fasp_sparse_rapcmp_
sparse_util.c, 555
fasp_sparse_rapms_
sparse_util.c, 556
fasp_sparse_wta_
sparse_util.c, 557
fasp_sparse_wtams_
sparse_util.c, 557
fasp_sparse_ytx_
sparse_util.c, 558
fasp_sparse_ytxbig_
sparse_util.c, 559
fasp_srtr
ilu.c, 251
fasp_symbfactor
ilu.c, 252
fasp_topological_sorting_ilu
ordering.c, 355
fasp_vector_read
io.c, 306
fasp_vector_write
io.c, 307
fasp_wrapper_dbsr_krylov_amg
wrapper.c, 597
fasp_wrapper_dcoo_dbsr_krylov_amg
wrapper.c, 598
fmgcycle.c, 226
fasp_solver_fmgcycle, 226
formats.c, 227
fasp_format_dblc_dcsr, 228
fasp_format_dbsr_dcoo, 228
fasp_format_dbsr_dcsr, 229
fasp_format_dcoo_dcsr, 229
fasp_format_dcsr_dbsr, 230
fasp_format_dcsr_dcoo, 230
fasp_format_dcsrl_dcsr, 231
fasp_format_dstr_dbsr, 232
fasp_format_dstr_dcsr, 232

G0PT
fasp_const.h, 208
GE
fasp.h, 186
givens.c, 233
fasp_aux_givens, 233
gmg_poisson.c, 234
fasp_poisson_fgmg_1D, 235
fasp_poisson_fgmg_2D, 236
fasp_poisson_fgmg_3D, 236
fasp_poisson_gmg_1D, 238
fasp_poisson_gmg_2D, 239
fasp_poisson_gmg_3D, 239
fasp_poisson_pcg_gmg_1D, 240
fasp_poisson_pcg_gmg_2D, 241
fasp_poisson_pcg_gmg_3D, 242

graphics.c, 243
    fasp_dbsr_plot, 243
    fasp_dbsr_subplot, 244
    fasp_dcsr_plot, 245
    fasp_dcsr_subplot, 245
    fasp_grid2d_plot, 246
grid2d, 38
    e, 39
    edges, 39
    ediri, 39
    efather, 39
    fasp.h, 191
    p, 39
    pdiri, 40
    pfather, 40
    s, 40
    t, 40
    tfather, 40
    triangles, 40
    vertices, 41
GT
    fasp.h, 187

ICNTL
    interface_mumps.c, 270
iCOOmat, 41
    fasp.h, 191
iCSRmat, 42
    fasp.h, 191
ILU_C_VERSION
    fasp.h, 187
ILU_MC_OMP
    fasp_const.h, 209
ILU_data, 43
ILU_droptol
    input_param, 52
ILU_lfil
    input_param, 52
ILU_param, 45
ILU_permtol
    input_param, 52
ILU_relax
    input_param, 53
ILU_type
    input_param, 53
ILUk
    fasp_const.h, 209
ILUt
    fasp_const.h, 209
ILUtp
    fasp_const.h, 209
IMAP
    fasp.h, 192
INTERP_DIR
    fasp_const.h, 209
INTERP_ENG
    fasp_const.h, 210
INTERP_STD
    fasp_const.h, 210
INT
    fasp.h, 187
ISNAN
    fasp.h, 187
ISPT
    fasp_const.h, 210
idenmat, 43
    fasp.h, 191
ilength
    io.c, 308
ilu.c, 247
    fasp_iluk, 248
    fasp_ilut, 248
    fasp_ilutp, 249
    fasp_qsplit, 251
    fasp_srtr, 251
    fasp_symbfactor, 252
ilu_setup_bsr.c, 255
    fasp_ilu_dbsr_setup, 256
    fasp_ilu_dbsr_setup_levsch_omp, 256
    fasp_ilu_dbsr_setup_mc_omp, 257
    fasp_ilu_dbsr_setup_omp, 258
ilu_setup_csr.c, 259
    fasp_ilu_dcsr_setup, 259
ilu_setup_str.c, 260
    fasp_ilu_dstr_setup0, 260
    fasp_ilu_dstr_setup1, 261
ilu_solve_omp
    smoother_bsr.c, 480
inifile
    input_param, 53
init.c, 262
    fasp_Schwarz_data_free, 267
    fasp_amg_data_bsr_create, 262
    fasp_amg_data_bsr_free, 263
    fasp_amg_data_create, 263
    fasp_amg_data_free, 264
    fasp_ilu_data_alloc, 264
    fasp_ilu_data_free, 265
    fasp_ilu_data_null, 265
    fasp_precond_data_null, 266
    fasp_precond_null, 266
input.c, 267
    fasp_param_check, 268
    fasp_param_input, 268
input_param, 45
    AMG_ILU_levels, 48
    AMG_Schwarz_levels, 50
    AMG_aggregation_type, 47

AMG_aggressive_level, 47
AMG_aggressive_path, 47
AMG_amli_degree, 47
AMG_coarse_dof, 47
AMG_coarse_scaling, 48
AMG_coarse_solver, 48
AMG_coarsening_type, 48
AMG_cycle_type, 48
AMG_interpolation_type, 48
AMG_levels, 49
AMG_max_aggregation, 49
AMG_max_row_sum, 49
AMG_maxit, 49
AMG_nl_amli_krylov_type, 49
AMG_pair_number, 49
AMG_polynomial_degree, 50
AMG_postsmooth_iter, 50
AMG_presmooth_iter, 50
AMG_quality_bound, 50
AMG_relaxation, 50
AMG_smooth_filter, 51
AMG_smooth_order, 51
AMG_smoother, 51
AMG_strong_coupled, 51
AMG_strong_threshold, 51
AMG_tentative_smooth, 51
AMG_tol, 52
AMG_truncation_threshold, 52
AMG_type, 52
ILU_droptol, 52
ILU_lfil, 52
ILU_permtol, 52
ILU_relax, 53
ILU_type, 53
inifile, 53
itsolver_maxit, 53
itsolver_tol, 53
output_type, 53
precond_type, 54
print_level, 54
problem_num, 54
restart, 54
Schwarz_blksolver, 54
Schwarz_maxlvl, 54
Schwarz_mmsize, 55
Schwarz_type, 55
solver_type, 55
stop_type, 55
workdir, 55
interface_mumps.c, 269
fasp_solver_mumps, 270
fasp_solver_mumps_steps, 270
ICNTL, 270
interface_pardiso.c, 272
fasp_solver_pardiso, 272
interface_samg.c, 273
dCSRmat2SAMGInput, 274
dvector2SAMGInput, 274
interface_superlu.c, 275
fasp_solver_superlu, 275
interface_umfpack.c, 276
fasp_solver_umfpack, 276
interpolation.c, 277
fasp_amg_interp, 278
fasp_amg_interp1, 278
fasp_amg_interp_trunc, 279
interpolation_em.c, 280
fasp_amg_interp_em, 280
io.c, 281
dlength, 308
fasp_dbsr_print, 283
fasp_dbsr_read, 284
fasp_dbsr_write, 284
fasp_dbsr_write_coo, 286
fasp_dcoo1_read, 286
fasp_dcoo_print, 287
fasp_dcoo_read, 288
fasp_dcoo_shift_read, 288
fasp_dcoo_write, 289
fasp_dcsr_print, 290
fasp_dcsr_read, 290
fasp_dcsr_write_coo, 291
fasp_dcsrvec1_read, 291
fasp_dcsrvec1_write, 292
fasp_dcsrvec2_read, 293
fasp_dcsrvec2_write, 294
fasp_dmtx_read, 295
fasp_dmtxsym_read, 295
fasp_dstr_print, 296
fasp_dstr_read, 296
fasp_dstr_write, 297
fasp_dvec_print, 298
fasp_dvec_read, 298
fasp_dvec_write, 299
fasp_dvecind_read, 299
fasp_dvecind_write, 300
fasp_hb_read, 301
fasp_ivec_print, 301
fasp_ivec_read, 302
fasp_ivec_write, 303
fasp_ivecind_read, 303
fasp_matrix_read, 304
fasp_matrix_read_bin, 305
fasp_matrix_write, 305
fasp_vector_read, 306
fasp_vector_write, 307
ilength, 308
itsolver_blc.c, 309

fasp_solver_dblc_itsolver, 309
fasp_solver_dblc_krylov, 310
fasp_solver_dblc_krylov_block_3, 311
fasp_solver_dblc_krylov_block_4, 311
fasp_solver_dblc_krylov_sweeping, 312
itsolver_bsr.c, 313
fasp_solver_dbsr_itsolver, 314
fasp_solver_dbsr_krylov, 314
fasp_solver_dbsr_krylov_amg, 315
fasp_solver_dbsr_krylov_amg_nk, 316
fasp_solver_dbsr_krylov_diag, 317
fasp_solver_dbsr_krylov_ilu, 317
fasp_solver_dbsr_krylov_nk_amg, 319
itsolver_csr.c, 320
fasp_solver_dcsr_itsolver, 321
fasp_solver_dcsr_krylov, 321
fasp_solver_dcsr_krylov_Schwarz, 326
fasp_solver_dcsr_krylov_amg, 322
fasp_solver_dcsr_krylov_amg_nk, 323
fasp_solver_dcsr_krylov_diag, 324
fasp_solver_dcsr_krylov_ilu, 324
fasp_solver_dcsr_krylov_ilu_M, 325
itsolver_maxit
input_param, 53
itsolver_mf.c, 327
fasp_solver_itsolver, 327
fasp_solver_itsolver_init, 328
fasp_solver_krylov, 329
itsolver_param, 56
itsolver_type, 56
maxit, 56
precond_type, 57
print_level, 57
restart, 57
stop_type, 57
tol, 57
itsolver_str.c, 330
fasp_solver_dstr_itsolver, 330
fasp_solver_dstr_krylov, 331
fasp_solver_dstr_krylov_blockgs, 331
fasp_solver_dstr_krylov_diag, 332
fasp_solver_dstr_krylov_ilu, 333
itsolver_tol
input_param, 53
itsolver_type
itsolver_param, 56
ivector, 58
fasp.h, 191

JA
dBSRmat, 33

LONGLONG
fasp.h, 188
LONG
fasp.h, 188
LU_diag
precond_block_data, 66
LU_P
precond_FASP_blkoil_data, 80
LU_S
precond_FASP_blkoil_data, 80
LE
fasp.h, 187
Link, 58
LinkList
fasp.h, 192
linked_list, 59
ListElement
fasp.h, 192
local_LU
precond_sweeping_data, 85
local_A
precond_sweeping_data, 85
local_index
precond_sweeping_data, 85
LS
fasp.h, 188
lu.c, 334
fasp_smat_lu_decomp, 334
fasp_smat_lu_solve, 335

MAT_BLC
fasp_const.h, 210
MAT_BSR
fasp_const.h, 211
MAT_CSRL
fasp_const.h, 211
MAT_CSR
fasp_const.h, 211
MAT_FREE
fasp_const.h, 211
MAT_STR
fasp_const.h, 211
MAT_SymCSR
fasp_const.h, 212
MAT_bBSR
fasp_const.h, 210
MAX_AMG_LVL
fasp_const.h, 212
MAX_CRATE
fasp_const.h, 212
MAX_REFINE_LVL
fasp_const.h, 212
MAX_RESTART
fasp_const.h, 212
MAX_STAG
fasp_const.h, 212
MAXIMAP

fasp.h, 193
MAX
fasp.h, 188
MIN_CDOF
fasp_const.h, 213
MIN_CRATE
fasp_const.h, 213
MIN
fasp.h, 188
mallinfo, 60
malloc_chunk, 60
malloc_params, 61
malloc_segment, 61
malloc_state, 62
malloc_tree_chunk, 62
maxit
itsolver_param, 56
precond_FASP_blkoil_data, 81
memory.c, 336
fasp_mem_calloc, 337
fasp_mem_check, 337
fasp_mem_dcsr_check, 338
fasp_mem_free, 338
fasp_mem_iludata_check, 339
fasp_mem_realloc, 340
fasp_mem_usage, 340
total_alloc_count, 341
total_alloc_mem, 341
message.c, 341
fasp_chkerr, 342
print_amgcomplexity, 342
print_amgcomplexity_bsr, 343
print_cputime, 343
print_itinfo, 344
print_message, 344
mgcycle.c, 345
fasp_solver_mgcycle, 345
fasp_solver_mgcycle_bsr, 346
mgl
precond_block_data, 67
mgl_data
precond_FASP_blkoil_data, 81
mgrecur.c, 346
fasp_solver_mgrecur, 347
Mumps_data, 63
mxv_matfree, 63

NEDMALLOC
fasp.h, 189
NL_AMLI_CYCLE
fasp_const.h, 213
NO_ORDER
fasp_const.h, 213
nedmallinfo, 64
neigh
precond_FASP_blkoil_data, 81
NumLayers
precond_sweeping_data, 85
nx_rb
fasp.h, 193
ny_rb
fasp.h, 193
nz_rb
fasp.h, 193

OFF
fasp_const.h, 213
OPENMP_HOLDS
fasp_const.h, 214
ON
fasp_const.h, 214
order
precond_FASP_blkoil_data, 81
precond_block_reservoir_data, 70
ordering.c, 348
fasp_BinarySearch, 353
fasp_aux_dQuickSort, 348
fasp_aux_dQuickSortIndex, 349
fasp_aux_iQuickSort, 350
fasp_aux_iQuickSortIndex, 350
fasp_aux_merge, 351
fasp_aux_msort, 352
fasp_aux_unique, 352
fasp_dcsr_CMK_order, 354
fasp_dcsr_RCMK_order, 354
fasp_multicolors_independent_set, 355
fasp_topological_sorting_ilu, 355
output_type
input_param, 53

p
grid2d, 39
PAIRWISE
fasp_const.h, 214
PREC_AMG
fasp_const.h, 214
PREC_DIAG
fasp_const.h, 214
PREC_FMG
fasp_const.h, 215
PREC_ILU
fasp_const.h, 215
PREC_NULL
fasp_const.h, 215
PREC_SCHWARZ
fasp_const.h, 215
PRINT_ALL
fasp_const.h, 215
PRINT_MIN

fasp_const.h, 216
PRINT_MORE
fasp_const.h, 216
PRINT_MOST
fasp_const.h, 216
PRINT_NONE
fasp_const.h, 216
PRINT_SOME
fasp_const.h, 216
PUT_INT
fasp.h, 189
PUT_REAL
fasp.h, 189
parameters.c, 356
fasp_param_Schwarz_init, 364
fasp_param_Schwarz_print, 364
fasp_param_Schwarz_set, 365
fasp_param_amg_init, 357
fasp_param_amg_print, 357
fasp_param_amg_set, 359
fasp_param_amg_to_prec, 359
fasp_param_amg_to_prec_bsr, 360
fasp_param_ilu_init, 360
fasp_param_ilu_print, 361
fasp_param_ilu_set, 361
fasp_param_init, 362
fasp_param_input_init, 362
fasp_param_prec_to_amg, 363
fasp_param_prec_to_amg_bsr, 363
fasp_param_set, 365
fasp_param_solver_init, 366
fasp_param_solver_print, 366
fasp_param_solver_set, 367
Pardiso_data, 64
pbcgs.c, 368
fasp_solver_dblc_pbcgs, 370
fasp_solver_dblc_pvbcgs, 370
fasp_solver_dbsr_pbcgs, 371
fasp_solver_dbsr_pvbcgs, 372
fasp_solver_dcsr_pbcgs, 373
fasp_solver_dcsr_pvbcgs, 374
fasp_solver_dstr_pbcgs, 374
fasp_solver_dstr_pvbcgs, 375
pbcgs_mf.c, 376
fasp_solver_pbcgs, 378
fasp_solver_pvbcgs, 379
pcg.c, 380
fasp_solver_dblc_pcg, 381
fasp_solver_dbsr_pcg, 382
fasp_solver_dcsr_pcg, 383
fasp_solver_dstr_pcg, 384
pcg_mf.c, 385
fasp_solver_pcg, 386
pcgrid2d
fasp.h, 192
pdiri
grid2d, 40
perf_idx
precond_FASP_blkoil_data, 81
precond_block_reservoir_data, 70
perf_neigh
precond_FASP_blkoil_data, 81
pfather
grid2d, 40
pgcg.c, 387
fasp_solver_dcsr_pgcg, 388
pgcg_mf.c, 388
fasp_solver_pgcg, 389
pgcr.c, 390
fasp_solver_dcsr_pgcr, 391
fasp_solver_dcsr_pgcr1, 392
pgmres.c, 393
fasp_solver_dblc_pgmres, 394
fasp_solver_dbsr_pgmres, 395
fasp_solver_dcsr_pgmres, 396
fasp_solver_dstr_pgmres, 397
pgmres_mf.c, 398
fasp_solver_pgmres, 398
pgrid2d
fasp.h, 192
pivot
precond_FASP_blkoil_data, 82
precond_block_reservoir_data, 70
pivot_S
precond_FASP_blkoil_data, 82
pivotS
precond_block_reservoir_data, 70
pminres.c, 399
fasp_solver_dblc_pminres, 401
fasp_solver_dcsr_pminres, 401
fasp_solver_dstr_pminres, 402
pminres_mf.c, 403
fasp_solver_pminres, 405
PP
precond_FASP_blkoil_data, 82
precond_block_reservoir_data, 70
precond, 65
precond_FASP_blkoil_data, 78
A, 80
diaginv, 80
diaginv_noscale, 80
diaginv_S, 80
LU_P, 80
LU_S, 80
maxit, 81
mgl_data, 81
neigh, 81
order, 81

perf_idx, 81
perf_neigh, 81
pivot, 82
pivot_S, 82
PP, 82
r, 82
restart, 82
RR, 82
scaled, 83
SS, 83
tol, 83
w, 83
WW, 83
precond_blc.c, 406
fasp_precond_block_SGS_3, 412
fasp_precond_block_SGS_3_amg, 412
fasp_precond_block_diag_3, 407
fasp_precond_block_diag_3_amg, 407
fasp_precond_block_diag_4, 408
fasp_precond_block_lower_3, 408
fasp_precond_block_lower_3_amg, 410
fasp_precond_block_lower_4, 410
fasp_precond_block_upper_3, 414
fasp_precond_block_upper_3_amg, 414
fasp_precond_sweeping, 416
precond_block_data, 65
A_diag, 66
Ablc, 66
amgparam, 66
LU_diag, 66
mgl, 67
r, 67
precond_block_reservoir_data, 67
diag, 69
diaginv, 69
diaginvS, 69
fasp_block.h, 197
order, 70
perf_idx, 70
pivot, 70
pivotS, 70
PP, 70
r, 70
RR, 71
scaled, 71
SS, 71
w, 71
WW, 71
precond_bsr.c, 416
fasp_precond_dbsr_amg, 417
fasp_precond_dbsr_amg_nk, 418
fasp_precond_dbsr_diag, 418
fasp_precond_dbsr_diag_nc2, 420
fasp_precond_dbsr_diag_nc3, 421
fasp_precond_dbsr_diag_nc5, 421
fasp_precond_dbsr_diag_nc7, 422
fasp_precond_dbsr_ilu, 423
fasp_precond_dbsr_ilu_levsch_omp, 424
fasp_precond_dbsr_ilu_mc_omp, 424
fasp_precond_dbsr_nl_amli, 425
precond_csr.c, 426
fasp_precond_Schwarz, 434
fasp_precond_amg, 426
fasp_precond_amg_nk, 427
fasp_precond_amli, 427
fasp_precond_diag, 429
fasp_precond_famg, 429
fasp_precond_free, 431
fasp_precond_ilu, 431
fasp_precond_ilu_backward, 432
fasp_precond_ilu_forward, 433
fasp_precond_nl_amli, 433
fasp_precond_setup, 434
precond_data, 72
precond_data_bsr, 73
precond_data_str, 75
precond_diagbsr, 77
precond_diagstr, 77
precond_str.c, 435
fasp_precond_dstr_blockgs, 436
fasp_precond_dstr_diag, 436
fasp_precond_dstr_ilu0, 437
fasp_precond_dstr_ilu0_backward, 437
fasp_precond_dstr_ilu0_forward, 438
fasp_precond_dstr_ilu1, 438
fasp_precond_dstr_ilu1_backward, 440
fasp_precond_dstr_ilu1_forward, 440
precond_sweeping_data, 84
A, 85
Ai, 85
local_LU, 85
local_A, 85
local_index, 85
NumLayers, 85
r, 86
w, 86
precond_type
input_param, 54
itsolver_param, 57
print_amgcomplexity
message.c, 342
print_amgcomplexity_bsr
message.c, 343
print_cputime
message.c, 343
print_itinfo
message.c, 344
print_level

input_param, 54
itsolver_param, 57
print_message
message.c, 344
problem_num
input_param, 54
pvfgmres.c, 442
fasp_solver_dblc_pvfgmres, 443
fasp_solver_dbsr_pvfgmres, 444
fasp_solver_dcsr_pvfgmres, 445
pvfgmres_mf.c, 446
fasp_solver_pvfgmres, 446
pvgmres.c, 447
fasp_solver_dblc_pvgmres, 448
fasp_solver_dbsr_pvgmres, 449
fasp_solver_dcsr_pvgmres, 449
fasp_solver_dstr_pvgmres, 450
pvgmres_mf.c, 451
fasp_solver_pvgmres, 452

quadrature.c, 453
fasp_gauss2d, 453
fasp_quad2d, 454

r
precond_FASP_blkoil_data, 82
precond_block_data, 67
precond_block_reservoir_data, 70
precond_sweeping_data, 86
REAL
fasp.h, 189
RS_C1
fasp.h, 189
rap.c, 455
fasp_blas_dcsr_rap2, 455
restart
input_param, 54
itsolver_param, 57
precond_FASP_blkoil_data, 82
RR
precond_FASP_blkoil_data, 82
precond_block_reservoir_data, 71

s
grid2d, 40
SA_AMG
fasp_const.h, 217
SCHWARZ_BACKWARD
fasp_const.h, 217
SCHWARZ_FORWARD
fasp_const.h, 217
SCHWARZ_SYMMETRIC
fasp_const.h, 217
SHORT
fasp.h, 190
SMALLREAL2
fasp_const.h, 218
SMALLREAL
fasp_const.h, 217
SMOOTHER_BLKOIL
fasp_block.h, 195
SMOOTHER_CG
fasp_const.h, 218
SMOOTHER_GSOR
fasp_const.h, 218
SMOOTHER_GS
fasp_const.h, 218
SMOOTHER_JACOBI
fasp_const.h, 218
SMOOTHER_L1DIAG
fasp_const.h, 218
SMOOTHER_POLY
fasp_const.h, 219
SMOOTHER_SGSOR
fasp_const.h, 219
SMOOTHER_SGS
fasp_const.h, 219
SMOOTHER_SOR
fasp_const.h, 219
SMOOTHER_SPETEN
fasp_block.h, 195
SMOOTHER_SSOR
fasp_const.h, 219
SOLVER_AMG
fasp_const.h, 219
SOLVER_BiCGstab
fasp_const.h, 220
SOLVER_CG
fasp_const.h, 220
SOLVER_DEFAULT
fasp_const.h, 220
SOLVER_FMG
fasp_const.h, 220
SOLVER_GCG
fasp_const.h, 220
SOLVER_GCR
fasp_const.h, 221
SOLVER_GMRES
fasp_const.h, 221
SOLVER_MUMPS
fasp_const.h, 221
SOLVER_MinRes
fasp_const.h, 221
SOLVER_PARDISO
fasp_const.h, 221
SOLVER_SBiCGstab
fasp_const.h, 221
SOLVER_SCG
fasp_const.h, 222

SOLVER_SGCG
fasp_const.h, 222
SOLVER_SGMRES
fasp_const.h, 222
SOLVER_SMinRes
fasp_const.h, 222
SOLVER_SUPERLU
fasp_const.h, 222
SOLVER_SVFGMRES
fasp_const.h, 222
SOLVER_SVGMRES
fasp_const.h, 223
SOLVER_UMFPACK
fasp_const.h, 223
SOLVER_VBiCGstab
fasp_const.h, 223
SOLVER_VFGMRES
fasp_const.h, 223
SOLVER_VGMRES
fasp_const.h, 223
STAG_RATIO
fasp_const.h, 223
STOP_MOD_REL_RES
fasp_const.h, 224
STOP_REL_PRECRES
fasp_const.h, 224
STOP_REL_RES
fasp_const.h, 224
SWAP
smat.c, 460
scaled
precond_FASP_blkoil_data, 83
precond_block_reservoir_data, 71
Schwarz_blksolver
input_param, 54
Schwarz_data, 86
Schwarz_maxlvl
input_param, 54
Schwarz_mmsize
input_param, 55
Schwarz_param, 88
schwarz_setup.c, 456
fasp_Schwarz_get_block_matrix, 457
fasp_Schwarz_setup, 458
fasp_dcsr_Schwarz_backward_smoother, 456
fasp_dcsr_Schwarz_forward_smoother, 457
Schwarz_type
input_param, 55
smat.c, 459
fasp_blas_smat_Linfinity, 464
fasp_blas_smat_inv, 460
fasp_blas_smat_inv_nc, 461
fasp_blas_smat_inv_nc2, 461
fasp_blas_smat_inv_nc3, 462
fasp_blas_smat_inv_nc4, 462
fasp_blas_smat_inv_nc5, 463
fasp_blas_smat_inv_nc7, 463
fasp_blas_smat_invp_nc, 464
fasp_iden_free, 465
fasp_smat_identity, 465
fasp_smat_identity_nc2, 466
fasp_smat_identity_nc3, 466
fasp_smat_identity_nc5, 467
fasp_smat_identity_nc7, 467
SWAP, 460
smoother_bsr.c, 468
fasp_smoother_dbsr_gs, 469
fasp_smoother_dbsr_gs1, 470
fasp_smoother_dbsr_gs_ascend, 470
fasp_smoother_dbsr_gs_ascend1, 471
fasp_smoother_dbsr_gs_descend, 472
fasp_smoother_dbsr_gs_descend1, 472
fasp_smoother_dbsr_gs_order1, 473
fasp_smoother_dbsr_gs_order2, 474
fasp_smoother_dbsr_ilu, 474
fasp_smoother_dbsr_jacobi, 475
fasp_smoother_dbsr_jacobi1, 475
fasp_smoother_dbsr_jacobi_setup, 476
fasp_smoother_dbsr_sor, 477
fasp_smoother_dbsr_sor1, 477
fasp_smoother_dbsr_sor_ascend, 478
fasp_smoother_dbsr_sor_descend, 479
fasp_smoother_dbsr_sor_order, 479
ilu_solve_omp, 480
smoother_csr.c, 481
fasp_smoother_dcsr_L1diag, 486
fasp_smoother_dcsr_gs, 482
fasp_smoother_dcsr_gs_cf, 482
fasp_smoother_dcsr_gs_rb3d, 483
fasp_smoother_dcsr_ilu, 484
fasp_smoother_dcsr_jacobi, 484
fasp_smoother_dcsr_kaczmarz, 485
fasp_smoother_dcsr_sgs, 486
fasp_smoother_dcsr_sor, 487
fasp_smoother_dcsr_sor_cf, 488
smoother_csr_cr.c, 489
fasp_smoother_dcsr_gscr, 489
smoother_csr_poly.c, 490
fasp_smoother_dcsr_poly, 491
fasp_smoother_dcsr_poly_old, 491
smoother_str.c, 493
fasp_generate_diaginv_block, 494
fasp_smoother_dstr_gs, 495
fasp_smoother_dstr_gs1, 495
fasp_smoother_dstr_gs_ascend, 496
fasp_smoother_dstr_gs_cf, 497
fasp_smoother_dstr_gs_descend, 497
fasp_smoother_dstr_gs_order, 499

fasp_smoother_dstr_jacobi, 499
fasp_smoother_dstr_jacobi1, 500
fasp_smoother_dstr_schwarz, 501
fasp_smoother_dstr_sor, 501
fasp_smoother_dstr_sor1, 502
fasp_smoother_dstr_sor_ascend, 503
fasp_smoother_dstr_sor_cf, 503
fasp_smoother_dstr_sor_descend, 505
fasp_smoother_dstr_sor_order, 506
solver_type
input_param, 55
sparse_block.c, 506
fasp_dblc_free, 507
fasp_dbsr_Linfinity_dcsr, 509
fasp_dbsr_getblk, 507
fasp_dbsr_getblk_dcsr, 508
fasp_dcsr_getblk, 509
sparse_bsr.c, 510
fasp_dbsr_alloc, 511
fasp_dbsr_cp, 512
fasp_dbsr_create, 512
fasp_dbsr_diagLU2, 517
fasp_dbsr_diagLU, 517
fasp_dbsr_diaginv, 513
fasp_dbsr_diaginv2, 514
fasp_dbsr_diaginv3, 514
fasp_dbsr_diaginv4, 516
fasp_dbsr_diagpref, 518
fasp_dbsr_free, 519
fasp_dbsr_getdiag, 519
fasp_dbsr_getdiaginv, 520
fasp_dbsr_null, 520
fasp_dbsr_perm, 521
fasp_dbsr_trans, 522
sparse_coo.c, 522
fasp_dcoo_alloc, 523
fasp_dcoo_create, 523
fasp_dcoo_free, 524
fasp_dcoo_shift, 525
sparse_csr.c, 525
fasp_dcsr_alloc, 527
fasp_dcsr_compress, 527
fasp_dcsr_compress_inplace, 528
fasp_dcsr_cp, 529
fasp_dcsr_create, 529
fasp_dcsr_diagpref, 530
fasp_dcsr_free, 531
fasp_dcsr_getcol, 531
fasp_dcsr_getdiag, 532
fasp_dcsr_multicoloring, 532
fasp_dcsr_null, 533
fasp_dcsr_perm, 533
fasp_dcsr_permz, 534
fasp_dcsr_regdiag, 535
fasp_dcsr_shift, 535
fasp_dcsr_sort, 536
fasp_dcsr_sortz, 536
fasp_dcsr_symdiagscale, 537
fasp_dcsr_sympat, 537
fasp_dcsr_trans, 538
fasp_dcsr_transz, 539
fasp_icsr_cp, 539
fasp_icsr_create, 540
fasp_icsr_free, 540
fasp_icsr_null, 542
fasp_icsr_trans, 542
sparse_csrl.c, 543
fasp_dcsrl_create, 544
fasp_dcsrl_free, 545
sparse_str.c, 545
fasp_dstr_alloc, 546
fasp_dstr_cp, 547
fasp_dstr_create, 547
fasp_dstr_free, 548
fasp_dstr_null, 548
sparse_util.c, 549
fasp_sparse_MIS, 554
fasp_sparse_aat_, 550
fasp_sparse_abyb_, 551
fasp_sparse_abybms_, 552
fasp_sparse_aplbms_, 552
fasp_sparse_aplusb_, 553
fasp_sparse_iit_, 554
fasp_sparse_rapcmp_, 555
fasp_sparse_rapms_, 556
fasp_sparse_wta_, 557
fasp_sparse_wtams_, 557
fasp_sparse_ytx_, 558
fasp_sparse_ytxbig_, 559
spbcgs.c, 559
fasp_solver_dblc_spbcgs, 561
fasp_solver_dbsr_spbcgs, 562
fasp_solver_dcsr_spbcgs, 563
fasp_solver_dstr_spbcgs, 564
spcg.c, 565
fasp_solver_dblc_spcg, 566
fasp_solver_dcsr_spcg, 567
fasp_solver_dstr_spcg, 568
spgmres.c, 569
fasp_solver_dblc_spgmres, 570
fasp_solver_dbsr_spgmres, 570
fasp_solver_dcsr_spgmres, 571
fasp_solver_dstr_spgmres, 572
spminres.c, 573
fasp_solver_dblc_spminres, 575
fasp_solver_dcsr_spminres, 576
fasp_solver_dstr_spminres, 577
spvgmres.c, 578

fasp_solver_dblc_spvgmres, 578
fasp_solver_dbsr_spvgmres, 580
fasp_solver_dcsr_spvgmres, 581
fasp_solver_dstr_spvgmres, 582
SS
precond_FASP_blkoil_data, 83
precond_block_reservoir_data, 71
stop_type
input_param, 55
itsolver_param, 57

t
grid2d, 40
THDs_AMG_GS
threads.c, 584
THDs_CPR_gGS
threads.c, 584
THDs_CPR_lGS
threads.c, 585
TRUE
fasp_const.h, 224
tfather
grid2d, 40
threads.c, 583
FASP_GET_START_END, 583
fasp_set_GS_threads, 584
THDs_AMG_GS, 584
THDs_CPR_gGS, 584
THDs_CPR_lGS, 585
timing.c, 585
fasp_gettime, 585
tol
itsolver_param, 57
precond_FASP_blkoil_data, 83
total_alloc_count
fasp.h, 193
memory.c, 341
total_alloc_mem
fasp.h, 193
memory.c, 341
triangles
grid2d, 40

UA_AMG
fasp_const.h, 224
UNPT
fasp_const.h, 225
USERDEFINED
fasp_const.h, 225

V_CYCLE
fasp_const.h, 225
VMB
fasp_const.h, 225
val
dBSRmat, 33
vec.c, 586
fasp_dvec_alloc, 587
fasp_dvec_cp, 587
fasp_dvec_create, 587
fasp_dvec_free, 588
fasp_dvec_isnan, 588
fasp_dvec_maxdiff, 589
fasp_dvec_null, 590
fasp_dvec_rand, 590
fasp_dvec_set, 591
fasp_dvec_symdiagscale, 592
fasp_ivec_alloc, 592
fasp_ivec_create, 593
fasp_ivec_free, 593
fasp_ivec_set, 594
vertices
grid2d, 41

w
precond_FASP_blkoil_data, 83
precond_block_reservoir_data, 71
precond_sweeping_data, 86
W_CYCLE
fasp_const.h, 225
workdir
input_param, 55
wrapper.c, 595
fasp_fwrapper_amg_, 595
fasp_fwrapper_krylov_amg_, 596
fasp_wrapper_dbsr_krylov_amg, 597
fasp_wrapper_dcoo_dbsr_krylov_amg, 598
WW
precond_FASP_blkoil_data, 83
precond_block_reservoir_data, 71