

Fast Auxiliary Space Preconditioning

1.6.8 Feb/08/2015

Generated by Doxygen 1.8.9.1

Sun Feb 8 2015 21:34:50

Contents

1	Introduction	1
2	How to obtain FASP	3
3	Building and Installation	5
4	Developers	7
5	Doxxygen	9
6	Data Structure Index	11
6.1	Data Structures	11
7	File Index	13
7.1	File List	13
8	Data Structure Documentation	19
8.1	AMG_data Struct Reference	19
8.1.1	Detailed Description	20
8.2	AMG_data_bsr Struct Reference	20
8.2.1	Detailed Description	22
8.3	AMG_param Struct Reference	22
8.3.1	Detailed Description	24
8.4	block_BSR Struct Reference	25
8.4.1	Detailed Description	25
8.5	block_dCSRmat Struct Reference	25
8.5.1	Detailed Description	26
8.6	block_dvector Struct Reference	26
8.6.1	Detailed Description	27
8.7	block_iCSRmat Struct Reference	27
8.7.1	Detailed Description	28

8.8	block_ivector Struct Reference	28
8.8.1	Detailed Description	29
8.9	block_Reservoir Struct Reference	29
8.9.1	Detailed Description	30
8.10	dBSRmat Struct Reference	30
8.10.1	Detailed Description	31
8.10.2	Field Documentation	31
8.10.2.1	JA	31
8.10.2.2	val	31
8.11	dCOOmat Struct Reference	32
8.11.1	Detailed Description	32
8.12	dCSRLmat Struct Reference	32
8.12.1	Detailed Description	33
8.13	dCSRmat Struct Reference	33
8.13.1	Detailed Description	34
8.14	ddenmat Struct Reference	34
8.14.1	Detailed Description	34
8.15	dSTRmat Struct Reference	34
8.15.1	Detailed Description	35
8.16	dvector Struct Reference	35
8.16.1	Detailed Description	36
8.17	grid2d Struct Reference	36
8.17.1	Detailed Description	36
8.17.2	Field Documentation	36
8.17.2.1	e	36
8.17.2.2	edges	37
8.17.2.3	ediri	37
8.17.2.4	efather	37
8.17.2.5	p	37
8.17.2.6	pdiri	37
8.17.2.7	pfather	37
8.17.2.8	s	37
8.17.2.9	t	37
8.17.2.10	tfather	37
8.17.2.11	triangles	38
8.17.2.12	vertices	38
8.18	iCOOmat Struct Reference	38

8.18.1 Detailed Description	38
8.19 iCSRmat Struct Reference	39
8.19.1 Detailed Description	39
8.20 idenmat Struct Reference	39
8.20.1 Detailed Description	40
8.21 ILU_data Struct Reference	40
8.21.1 Detailed Description	40
8.22 ILU_param Struct Reference	40
8.22.1 Detailed Description	41
8.23 input_param Struct Reference	41
8.23.1 Detailed Description	42
8.23.2 Field Documentation	42
8.23.2.1 AMG_aggregation_type	42
8.23.2.2 AMG_aggressive_level	43
8.23.2.3 AMG_aggressive_path	43
8.23.2.4 AMG_amli_degree	43
8.23.2.5 AMG_coarse_dof	43
8.23.2.6 AMG_coarse_scaling	43
8.23.2.7 AMG_coarse_solver	43
8.23.2.8 AMG_coarsening_type	43
8.23.2.9 AMG_cycle_type	43
8.23.2.10 AMG_ILU_levels	43
8.23.2.11 AMG_interpolation_type	44
8.23.2.12 AMG_levels	44
8.23.2.13 AMG_max_aggregation	44
8.23.2.14 AMG_max_row_sum	44
8.23.2.15 AMG_maxit	44
8.23.2.16 AMG_nl_amli_krylov_type	44
8.23.2.17 AMG_pair_number	44
8.23.2.18 AMG_polynomial_degree	44
8.23.2.19 AMG_postsMOOTH_iter	44
8.23.2.20 AMG_presMOOTH_iter	45
8.23.2.21 AMG_relaxation	45
8.23.2.22 AMG_Schwarz_levels	45
8.23.2.23 AMG_smooth_filter	45
8.23.2.24 AMG_smooth_order	45
8.23.2.25 AMG_smoothen	45

8.23.2.26 AMG_strong_coupled	45
8.23.2.27 AMG_strong_threshold	45
8.23.2.28 AMG_tentative_smooth	45
8.23.2.29 AMG_tol	46
8.23.2.30 AMG_truncation_threshold	46
8.23.2.31 AMG_type	46
8.23.2.32 ILU_droptol	46
8.23.2.33 ILU_lfil	46
8.23.2.34 ILU_permtol	46
8.23.2.35 ILU_relax	46
8.23.2.36 ILU_type	46
8.23.2.37 infile	46
8.23.2.38 itsolver_maxit	47
8.23.2.39 itsolver_tol	47
8.23.2.40 output_type	47
8.23.2.41 precond_type	47
8.23.2.42 print_level	47
8.23.2.43 problem_num	47
8.23.2.44 restart	47
8.23.2.45 Schwarz_blksolver	47
8.23.2.46 Schwarz_maxlvl	47
8.23.2.47 Schwarz_mmsize	48
8.23.2.48 Schwarz_type	48
8.23.2.49 solver_type	48
8.23.2.50 stop_type	48
8.23.2.51 workdir	48
8.24 itsolver_param Struct Reference	48
8.24.1 Detailed Description	49
8.24.2 Field Documentation	49
8.24.2.1 itsolver_type	49
8.24.2.2 maxit	49
8.24.2.3 precond_type	49
8.24.2.4 print_level	49
8.24.2.5 restart	49
8.24.2.6 stop_type	49
8.24.2.7 tol	49
8.25 ivector Struct Reference	50

8.25.1	Detailed Description	50
8.26	Link Struct Reference	50
8.26.1	Detailed Description	50
8.27	linked_list Struct Reference	51
8.27.1	Detailed Description	51
8.28	Mumps_data Struct Reference	51
8.28.1	Detailed Description	52
8.29	mxv_matfree Struct Reference	52
8.29.1	Detailed Description	52
8.30	precond Struct Reference	52
8.30.1	Detailed Description	53
8.31	precond_block_data Struct Reference	53
8.31.1	Detailed Description	54
8.31.2	Field Documentation	54
8.31.2.1	A_diag	54
8.31.2.2	Abcsr	54
8.31.2.3	amgparam	54
8.31.2.4	LU_diag	54
8.31.2.5	mgl	54
8.31.2.6	r	54
8.32	precond_block_reservoir_data Struct Reference	55
8.32.1	Detailed Description	57
8.32.2	Field Documentation	57
8.32.2.1	diag	57
8.32.2.2	diaginv	57
8.32.2.3	diaginvS	57
8.32.2.4	order	57
8.32.2.5	perf_idx	57
8.32.2.6	pivot	57
8.32.2.7	pivotS	57
8.32.2.8	PP	58
8.32.2.9	r	58
8.32.2.10	RR	58
8.32.2.11	scaled	58
8.32.2.12	SS	58
8.32.2.13	w	58
8.32.2.14	WW	58

8.33 precond_data Struct Reference	58
8.33.1 Detailed Description	60
8.34 precond_data_bsr Struct Reference	60
8.34.1 Detailed Description	62
8.35 precond_data_str Struct Reference	62
8.35.1 Detailed Description	64
8.36 precond_diagbsr Struct Reference	64
8.36.1 Detailed Description	65
8.37 precond_diagstr Struct Reference	65
8.37.1 Detailed Description	66
8.38 precond_FASP_blkoil_data Struct Reference	66
8.38.1 Detailed Description	68
8.38.2 Field Documentation	69
8.38.2.1 A	69
8.38.2.2 diaginv	69
8.38.2.3 diaginv_noscale	69
8.38.2.4 diaginv_S	69
8.38.2.5 maxit	69
8.38.2.6 mgl_data	69
8.38.2.7 neigh	69
8.38.2.8 order	69
8.38.2.9 perf_idx	70
8.38.2.10 perf_neigh	70
8.38.2.11 pivot	70
8.38.2.12 pivot_S	70
8.38.2.13 PP	70
8.38.2.14 r	70
8.38.2.15 restart	70
8.38.2.16 RR	70
8.38.2.17 scaled	70
8.38.2.18 SS	71
8.38.2.19 tol	71
8.38.2.20 w	71
8.38.2.21 WW	71
8.39 precond_sweeping_data Struct Reference	71
8.39.1 Detailed Description	72
8.39.2 Field Documentation	73

8.39.2.1	A	73
8.39.2.2	Ai	73
8.39.2.3	local_A	73
8.39.2.4	local_index	73
8.39.2.5	local_LU	73
8.39.2.6	NumLayers	73
8.39.2.7	r	73
8.39.2.8	w	73
8.40	Schwarz_data Struct Reference	74
8.40.1	Detailed Description	75
8.41	Schwarz_param Struct Reference	75
8.41.1	Detailed Description	75
9	File Documentation	77
9.1	amg.c File Reference	77
9.1.1	Detailed Description	78
9.1.2	Function Documentation	78
9.1.2.1	fasp_solver_amg	78
9.2	amg_setup_cr.c File Reference	79
9.2.1	Detailed Description	80
9.2.2	Function Documentation	80
9.2.2.1	fasp_amg_setup_cr	80
9.3	amg_setup_rs.c File Reference	81
9.3.1	Detailed Description	82
9.3.2	Function Documentation	82
9.3.2.1	fasp_amg_setup_rs	82
9.4	amg_setup_sa.c File Reference	84
9.4.1	Detailed Description	85
9.4.2	Function Documentation	85
9.4.2.1	fasp_amg_setup_sa	85
9.4.2.2	fasp_amg_setup_sa_bsr	86
9.5	amg_setup_ua.c File Reference	86
9.5.1	Detailed Description	87
9.5.2	Function Documentation	87
9.5.2.1	fasp_amg_setup_ua	87
9.5.2.2	fasp_amg_setup_ua_bsr	87
9.6	amg_solve.c File Reference	88

9.6.1	Detailed Description	89
9.6.2	Function Documentation	89
9.6.2.1	<code>fasp_amg_solve</code>	89
9.6.2.2	<code>fasp_amg_solve_amli</code>	90
9.6.2.3	<code>fasp_amg_solve_nl_amli</code>	92
9.6.2.4	<code>fasp_famg_solve</code>	93
9.7	amlirecur.c File Reference	94
9.7.1	Detailed Description	95
9.7.2	Function Documentation	95
9.7.2.1	<code>fasp_amg_amli_coef</code>	95
9.7.2.2	<code>fasp_solver_amli</code>	96
9.7.2.3	<code>fasp_solver_nl_amli</code>	97
9.7.2.4	<code>fasp_solver_nl_amli_bsr</code>	99
9.8	array.c File Reference	101
9.8.1	Detailed Description	102
9.8.2	Function Documentation	102
9.8.2.1	<code>fasp_array_cp</code>	102
9.8.2.2	<code>fasp_array_cp_nc3</code>	102
9.8.2.3	<code>fasp_array_cp_nc5</code>	102
9.8.2.4	<code>fasp_array_cp_nc7</code>	103
9.8.2.5	<code>fasp_array_null</code>	103
9.8.2.6	<code>fasp_array_set</code>	104
9.8.2.7	<code>fasp_iarray_cp</code>	104
9.8.2.8	<code>fasp_iarray_set</code>	105
9.9	blas_array.c File Reference	105
9.9.1	Detailed Description	106
9.9.2	Function Documentation	106
9.9.2.1	<code>fasp_blas_array_ax</code>	106
9.9.2.2	<code>fasp_blas_array_axpby</code>	107
9.9.2.3	<code>fasp_blas_array_axpy</code>	108
9.9.2.4	<code>fasp_blas_array_axpyz</code>	109
9.9.2.5	<code>fasp_blas_array_dotprod</code>	109
9.9.2.6	<code>fasp_blas_array_norm1</code>	110
9.9.2.7	<code>fasp_blas_array_norm2</code>	110
9.9.2.8	<code>fasp_blas_array_norminf</code>	111
9.10	blas_bcsr.c File Reference	112
9.10.1	Detailed Description	112

9.10.2 Function Documentation	112
9.10.2.1 fasp blas bdbsr_aAxpy	113
9.10.2.2 fasp blas bdbsr_mxv	114
9.10.2.3 fasp blas bdcsr_aAxpy	115
9.10.2.4 fasp blas bdcsr_mxv	116
9.11 blas_bsr.c File Reference	116
9.11.1 Detailed Description	117
9.11.2 Function Documentation	118
9.11.2.1 fasp blas dbsr_aAxpby	118
9.11.2.2 fasp blas dbsr_aAxpy	119
9.11.2.3 fasp blas dbsr_aAxpy_agg	120
9.11.2.4 fasp blas dbsr_axm	121
9.11.2.5 fasp blas dbsr_mxm	122
9.11.2.6 fasp blas dbsr_mxv	123
9.11.2.7 fasp blas dbsr_mxv_agg	124
9.11.2.8 fasp blas dbsr_rap	125
9.11.2.9 fasp blas dbsr_rap1	126
9.11.2.10 fasp blas dbsr_rap_agg	127
9.12 blas_csr.c File Reference	128
9.12.1 Detailed Description	129
9.12.2 Function Documentation	129
9.12.2.1 fasp blas dcsr_aAxpy	129
9.12.2.2 fasp blas dcsr_aAxpy_agg	130
9.12.2.3 fasp blas dcsr_add	131
9.12.2.4 fasp blas dcsr_axm	132
9.12.2.5 fasp blas dcsr_mxm	132
9.12.2.6 fasp blas dcsr_mxv	133
9.12.2.7 fasp blas dcsr_mxv_agg	134
9.12.2.8 fasp blas dcsr_ptap	135
9.12.2.9 fasp blas dcsr_rap	136
9.12.2.10 fasp blas dcsr_rap4	137
9.12.2.11 fasp blas dcsr_rap_agg	138
9.12.2.12 fasp blas dcsr_rap_agg1	139
9.12.2.13 fasp blas dcsr_vmv	140
9.13 blas_csrl.c File Reference	140
9.13.1 Detailed Description	141
9.13.2 Function Documentation	141

9.13.2.1 fasp blas dcsrl_mxv	141
9.14 blas_smat.c File Reference	142
9.14.1 Detailed Description	144
9.14.2 Function Documentation	144
9.14.2.1 fasp blas array axpy_nc2	144
9.14.2.2 fasp blas array axpy_nc3	144
9.14.2.3 fasp blas array axpy_nc5	144
9.14.2.4 fasp blas array axpy_nc7	145
9.14.2.5 fasp blas array axpyz_nc2	145
9.14.2.6 fasp blas array axpyz_nc3	146
9.14.2.7 fasp blas array axpyz_nc5	146
9.14.2.8 fasp blas array axpyz_nc7	147
9.14.2.9 fasp blas smat_aAxpby	148
9.14.2.10 fasp blas smat_add	148
9.14.2.11 fasp blas smat_axm	149
9.14.2.12 fasp blas smat_mul	149
9.14.2.13 fasp blas smat_mul_nc2	150
9.14.2.14 fasp blas smat_mul_nc3	150
9.14.2.15 fasp blas smat_mul_nc5	151
9.14.2.16 fasp blas smat_mul_nc7	151
9.14.2.17 fasp blas smat_mxv	152
9.14.2.18 fasp blas smat_mxv_nc2	153
9.14.2.19 fasp blas smat_mxv_nc3	154
9.14.2.20 fasp blas smat_mxv_nc5	154
9.14.2.21 fasp blas smat_mxv_nc7	154
9.14.2.22 fasp blas smat_ymAx	155
9.14.2.23 fasp blas smat_ymAx_nc2	155
9.14.2.24 fasp blas smat_ymAx_nc3	156
9.14.2.25 fasp blas smat_ymAx_nc5	156
9.14.2.26 fasp blas smat_ymAx_nc7	156
9.14.2.27 fasp blas smat_ymAx_ns	157
9.14.2.28 fasp blas smat_ymAx_ns2	157
9.14.2.29 fasp blas smat_ymAx_ns3	158
9.14.2.30 fasp blas smat_ymAx_ns5	158
9.14.2.31 fasp blas smat_ymAx_ns7	159
9.14.2.32 fasp blas smat_ypAx	159
9.14.2.33 fasp blas smat_ypAx_nc2	160

9.14.2.34 fasp_blas_smat_ypAx_nc3	160
9.14.2.35 fasp_blas_smat_ypAx_nc5	161
9.14.2.36 fasp_blas_smat_ypAx_nc7	162
9.15 blas_str.c File Reference	162
9.15.1 Detailed Description	163
9.15.2 Function Documentation	163
9.15.2.1 fasp_blas_dstr_aApxy	163
9.15.2.2 fasp_blas_dstr_mxv	164
9.15.2.3 fasp_dstr_diagscale	164
9.16 blas_vec.c File Reference	165
9.16.1 Detailed Description	166
9.16.2 Function Documentation	166
9.16.2.1 fasp_blas_dvec_axpy	166
9.16.2.2 fasp_blas_dvec_axpyz	167
9.16.2.3 fasp_blas_dvec_dotprod	168
9.16.2.4 fasp_blas_dvec_norm1	168
9.16.2.5 fasp_blas_dvec_norm2	169
9.16.2.6 fasp_blas_dvec_norminf	169
9.16.2.7 fasp_blas_dvec_relerr	170
9.17 checkmat.c File Reference	171
9.17.1 Detailed Description	171
9.17.2 Function Documentation	172
9.17.2.1 fasp_check_dCSRmat	172
9.17.2.2 fasp_check_diagdom	173
9.17.2.3 fasp_check_diagpos	174
9.17.2.4 fasp_check_diagzero	175
9.17.2.5 fasp_check_iCSRmat	175
9.17.2.6 fasp_check_symm	175
9.18 coarsening_cr.c File Reference	176
9.18.1 Detailed Description	177
9.18.2 Function Documentation	177
9.18.2.1 fasp_amg_coarsening_cr	177
9.19 coarsening_rs.c File Reference	178
9.19.1 Detailed Description	179
9.19.2 Function Documentation	179
9.19.2.1 fasp_amg_coarsening_rs	179
9.20 convert.c File Reference	180

9.20.1	Detailed Description	181
9.20.2	Function Documentation	181
9.20.2.1	<code>endian_convert_int</code>	181
9.20.2.2	<code>endian_convert_real</code>	182
9.20.2.3	<code>fasp_aux_bbyteTodouble</code>	182
9.20.2.4	<code>fasp_aux_change_endian4</code>	183
9.20.2.5	<code>fasp_aux_change_endian8</code>	183
9.21	doxygen.h File Reference	184
9.21.1	Detailed Description	184
9.22	eigen.c File Reference	184
9.22.1	Detailed Description	185
9.22.2	Function Documentation	185
9.22.2.1	<code>fasp_dcsr_eig</code>	185
9.23	factor.f File Reference	186
9.23.1	Detailed Description	186
9.24	famg.c File Reference	186
9.24.1	Detailed Description	187
9.24.2	Function Documentation	187
9.24.2.1	<code>fasp_solver_famg</code>	187
9.25	fasp.h File Reference	188
9.25.1	Detailed Description	191
9.25.2	Macro Definition Documentation	192
9.25.2.1	<code>__FASP_HEADER__</code>	192
9.25.2.2	<code>ABS</code>	192
9.25.2.3	<code>DIAGONAL_PREF</code>	192
9.25.2.4	<code>DLMALLOC</code>	192
9.25.2.5	<code>FASP_GSRB</code>	192
9.25.2.6	<code>FASP_USE_ILU</code>	192
9.25.2.7	<code>GE</code>	192
9.25.2.8	<code>GT</code>	192
9.25.2.9	<code>INT</code>	193
9.25.2.10	<code>ISNAN</code>	193
9.25.2.11	<code>LE</code>	193
9.25.2.12	<code>LONG</code>	193
9.25.2.13	<code>LONGLONG</code>	193
9.25.2.14	<code>LS</code>	193
9.25.2.15	<code>MAX</code>	193

9.25.2.16 MIN	193
9.25.2.17 NEDMALLOC	194
9.25.2.18 PRT_INT	194
9.25.2.19 PRT_REAL	194
9.25.2.20 REAL	194
9.25.2.21 RS_C1	194
9.25.2.22 SHORT	194
9.25.3 TYPEDOC Documentation	194
9.25.3.1 dCOOmat	194
9.25.3.2 dCSRLmat	194
9.25.3.3 dCSRmat	195
9.25.3.4 ddenmat	195
9.25.3.5 dSTRmat	195
9.25.3.6 dvector	195
9.25.3.7 grid2d	195
9.25.3.8 iCOOmat	195
9.25.3.9 iCSRmat	195
9.25.3.10 idenmat	195
9.25.3.11 ivector	195
9.25.3.12 LinkList	195
9.25.3.13 ListElement	195
9.25.3.14 pcgrid2d	196
9.25.3.15 pgrid2d	196
9.25.4 VARIABLE Documentation	196
9.25.4.1 count	196
9.25.4.2 IMAP	196
9.25.4.3 MAXIMAP	196
9.25.4.4 nx_rb	196
9.25.4.5 ny_rb	196
9.25.4.6 nz_rb	196
9.25.4.7 total_alloc_count	196
9.25.4.8 total_alloc_mem	197
9.26 fasp_block.h File Reference	197
9.26.1 Detailed Description	198
9.26.2 TYPEDOC Documentation	199
9.26.2.1 block_BSR	199
9.26.2.2 block_dCSRmat	199

9.26.2.3	block_dvector	199
9.26.2.4	block_iCSRmat	199
9.26.2.5	block_ivector	199
9.26.2.6	block_Reservoir	199
9.26.2.7	dBSRmat	199
9.26.2.8	precond_block_reservoir_data	199
9.27	fasp_const.h File Reference	199
9.27.1	Detailed Description	203
9.27.2	Macro Definition Documentation	203
9.27.2.1	AMLI_CYCLE	203
9.27.2.2	ASCEND	203
9.27.2.3	BIGREAL	203
9.27.2.4	CF_ORDER	203
9.27.2.5	CGPT	203
9.27.2.6	CLASSIC_AMG	204
9.27.2.7	COARSE_AC	204
9.27.2.8	COARSE_CR	204
9.27.2.9	COARSE_MIS	204
9.27.2.10	COARSE_RS	204
9.27.2.11	CPFIRST	204
9.27.2.12	DESCEND	204
9.27.2.13	ERROR_ALLOC_MEM	204
9.27.2.14	ERROR_AMG_COARSE_TYPE	205
9.27.2.15	ERROR_AMG_COARSEING	205
9.27.2.16	ERROR_AMG_INTERP_TYPE	205
9.27.2.17	ERROR_AMG_SMOOTH_TYPE	205
9.27.2.18	ERROR_DATA_STRUCTURE	205
9.27.2.19	ERROR_DATA_ZERODIAG	205
9.27.2.20	ERROR_DUMMY_VAR	205
9.27.2.21	ERROR_INPUT_PAR	205
9.27.2.22	ERROR_LIC_TYPE	205
9.27.2.23	ERROR_MAT_SIZE	206
9.27.2.24	ERROR_MISC	206
9.27.2.25	ERROR_NUM_BLOCKS	206
9.27.2.26	ERROR_OPEN_FILE	206
9.27.2.27	ERROR_QUAD_DIM	206
9.27.2.28	ERROR_QUAD_TYPE	206

9.27.2.29 ERROR_REGRESS	206
9.27.2.30 ERROR_SOLVER_EXIT	206
9.27.2.31 ERROR_SOLVER_ILUSETUP	206
9.27.2.32 ERROR_SOLVER_MAXIT	207
9.27.2.33 ERROR_SOLVER_MISC	207
9.27.2.34 ERROR_SOLVER_PRECTYPE	207
9.27.2.35 ERROR_SOLVER_SOLSTAG	207
9.27.2.36 ERROR_SOLVER_STAG	207
9.27.2.37 ERROR_SOLVER_TOLSMALL	207
9.27.2.38 ERROR_SOLVER_TYPE	207
9.27.2.39 ERROR_UNKNOWN	207
9.27.2.40 ERROR_WRONG_FILE	207
9.27.2.41 FALSE	208
9.27.2.42 FASP_SUCCESS	208
9.27.2.43 FGPT	208
9.27.2.44 FPFIRST	208
9.27.2.45 G0PT	208
9.27.2.46 ILUk	208
9.27.2.47 ILUt	208
9.27.2.48 ILUtp	208
9.27.2.49 INTERP_DIR	209
9.27.2.50 INTERP_ENG	209
9.27.2.51 INTERP_STD	209
9.27.2.52 ISPT	209
9.27.2.53 MAT_bBSR	209
9.27.2.54 MAT_bCSR	209
9.27.2.55 MAT_BSR	209
9.27.2.56 MAT_CSR	209
9.27.2.57 MAT_CSRL	210
9.27.2.58 MAT_FREE	210
9.27.2.59 MAT_STR	210
9.27.2.60 MAT_SymCSR	210
9.27.2.61 MAX_AMG_LVL	210
9.27.2.62 MAX_CRATE	210
9.27.2.63 MAX_REFINE_LVL	210
9.27.2.64 MAX_RESTART	210
9.27.2.65 MAX_STAG	211

9.27.2.66 MIN_CDOF	211
9.27.2.67 MIN_CRATE	211
9.27.2.68 NL_AMLI_CYCLE	211
9.27.2.69 NO_ORDER	211
9.27.2.70 OFF	211
9.27.2.71 ON	211
9.27.2.72 OPENMP HOLDS	211
9.27.2.73 PAIRWISE	212
9.27.2.74 PREC_AMG	212
9.27.2.75 PREC_DIAG	212
9.27.2.76 PREC_FMGM	212
9.27.2.77 PREC_ILU	212
9.27.2.78 PREC_NULL	212
9.27.2.79 PREC_SCHWARZ	212
9.27.2.80 PRINT_ALL	212
9.27.2.81 PRINT_MIN	213
9.27.2.82 PRINT_MORE	213
9.27.2.83 PRINT_MOST	213
9.27.2.84 PRINT_NONE	213
9.27.2.85 PRINT_SOME	213
9.27.2.86 SA_AMG	213
9.27.2.87 SMALLREAL	213
9.27.2.88 SMOOTHBLKOIL	213
9.27.2.89 SMOOTHCG	214
9.27.2.90 SMOOTHGS	214
9.27.2.91 SMOOTHGSOR	214
9.27.2.92 SMOOTHJACOBI	214
9.27.2.93 SMOOTHL1DIAG	214
9.27.2.94 SMOOTHPOLY	214
9.27.2.95 SMOOTHSGS	214
9.27.2.96 SMOOTHSGSOR	214
9.27.2.97 SMOOTHSOR	215
9.27.2.98 SMOOTHSPETEN	215
9.27.2.99 SMOOTHSSOR	215
9.27.2.100 SOLVER_AMG	215
9.27.2.101 SOLVER_BiCGstab	215
9.27.2.102 SOLVERCG	215

9.27.2.103	SOLVER_DEFAULT	215
9.27.2.104	SOLVER_FM	215
9.27.2.105	SOLVER_GCG	216
9.27.2.106	SOLVER_GCR	216
9.27.2.107	SOLVER_GMRES	216
9.27.2.108	SOLVER_MinRes	216
9.27.2.109	SOLVER_MUMPS	216
9.27.2.110	SOLVER_SBiCGstab	216
9.27.2.111	SOLVER_SCG	216
9.27.2.112	SOLVER_SCGC	216
9.27.2.113	SOLVER_SGMRES	216
9.27.2.114	SOLVER_SMinRes	217
9.27.2.115	SOLVER_SUPERLU	217
9.27.2.116	SOLVER_SVFGMRES	217
9.27.2.117	SOLVER_SVGMRES	217
9.27.2.118	SOLVER_UMFPACK	217
9.27.2.119	SOLVER_VFGMRES	217
9.27.2.120	SOLVER_VGMRES	217
9.27.2.121	STAG_RATIO	217
9.27.2.122	STOP_MOD_REL_RES	217
9.27.2.123	STOP_REL_PRECRES	218
9.27.2.124	STOP_REL_RES	218
9.27.2.125	TRUE	218
9.27.2.126	UA_AMG	218
9.27.2.127	UNPT	218
9.27.2.128	USERDEFINED	218
9.27.2.129	V_CYCLE	218
9.27.2.130	VMB	218
9.27.2.131	W_CYCLE	219
9.28	fmgcycle.c File Reference	219
9.28.1	Detailed Description	219
9.28.2	Function Documentation	219
9.28.2.1	fasp_solver_fmgcycle	219
9.29	formats.c File Reference	220
9.29.1	Detailed Description	221
9.29.2	Function Documentation	222
9.29.2.1	fasp_format_bdcsrc_dscr	222

9.29.2.2 fasp_format_dbsr_dcoo	222
9.29.2.3 fasp_format_dbsr_dcsr	223
9.29.2.4 fasp_format_dcoo_dcsr	224
9.29.2.5 fasp_format_dcsr_dbsr	225
9.29.2.6 fasp_format_dcsr_dcoo	226
9.29.2.7 fasp_format_dcsrl_dcsr	227
9.29.2.8 fasp_format_dstr_dbsr	227
9.29.2.9 fasp_format_dstr_dcsr	228
9.30 givens.c File Reference	229
9.30.1 Detailed Description	229
9.30.2 Function Documentation	230
9.30.2.1 fasp_aux_givens	230
9.31 gmg_poisson.c File Reference	230
9.31.1 Detailed Description	231
9.31.2 Function Documentation	231
9.31.2.1 fasp_poisson_fgmg_1D	231
9.31.2.2 fasp_poisson_fgmg_2D	232
9.31.2.3 fasp_poisson_fgmg_3D	233
9.31.2.4 fasp_poisson_gmg_1D	234
9.31.2.5 fasp_poisson_gmg_2D	235
9.31.2.6 fasp_poisson_gmg_3D	236
9.31.2.7 fasp_poisson_pcg_gmg_1D	237
9.31.2.8 fasp_poisson_pcg_gmg_2D	238
9.31.2.9 fasp_poisson_pcg_gmg_3D	239
9.32 graphics.c File Reference	240
9.32.1 Detailed Description	241
9.32.2 Function Documentation	241
9.32.2.1 fasp_dbsr_plot	241
9.32.2.2 fasp_dbsr_subplot	242
9.32.2.3 fasp_dcsr_plot	243
9.32.2.4 fasp_dcsr_subplot	244
9.32.2.5 fasp_grid2d_plot	245
9.33 ilu.f File Reference	245
9.33.1 Detailed Description	246
9.34 ilu_setup_bsr.c File Reference	246
9.34.1 Detailed Description	247
9.34.2 Function Documentation	247

9.34.2.1 fasp_ilu_dbsr_setup	247
9.35 ilu_setup_csr.c File Reference	247
9.35.1 Detailed Description	248
9.35.2 Function Documentation	248
9.35.2.1 fasp_ilu_dcsr_setup	248
9.36 ilu_setup_str.c File Reference	249
9.36.1 Detailed Description	250
9.36.2 Function Documentation	250
9.36.2.1 fasp_ilu_dstr_setup0	250
9.36.2.2 fasp_ilu_dstr_setup1	251
9.37 init.c File Reference	253
9.37.1 Detailed Description	254
9.37.2 Function Documentation	255
9.37.2.1 fasp_amg_data_bsr_create	255
9.37.2.2 fasp_amg_data_bsr_free	255
9.37.2.3 fasp_amg_data_create	256
9.37.2.4 fasp_amg_data_free	257
9.37.2.5 fasp_ilu_data_alloc	258
9.37.2.6 fasp_ilu_data_free	258
9.37.2.7 fasp_ilu_data_null	259
9.37.2.8 fasp_precond_data_null	259
9.37.2.9 fasp_precond_null	260
9.37.2.10 fasp_Schwarz_data_free	261
9.38 input.c File Reference	261
9.38.1 Detailed Description	262
9.38.2 Function Documentation	262
9.38.2.1 fasp_param_check	262
9.38.2.2 fasp_param_input	263
9.39 interface_mumps.c File Reference	263
9.39.1 Detailed Description	264
9.39.2 Function Documentation	264
9.39.2.1 fasp_solver_mumps	264
9.39.2.2 fasp_solver_mumps_steps	265
9.40 interface_samg.c File Reference	265
9.40.1 Detailed Description	266
9.40.2 Function Documentation	266
9.40.2.1 dCSRmat2SAMGInput	266

9.40.2.2 <code>dvector2SAMGInput</code>	267
9.41 <code>interface_superlu.c</code> File Reference	268
9.41.1 Detailed Description	268
9.41.2 Function Documentation	268
9.41.2.1 <code>fasp_solver_superlu</code>	268
9.42 <code>interface_umfpack.c</code> File Reference	269
9.42.1 Detailed Description	270
9.42.2 Function Documentation	270
9.42.2.1 <code>fasp_solver_umfpack</code>	270
9.43 <code>interpolation.c</code> File Reference	271
9.43.1 Detailed Description	271
9.43.2 Function Documentation	272
9.43.2.1 <code>fasp_amg_interp</code>	272
9.43.2.2 <code>fasp_amg_interp1</code>	272
9.43.2.3 <code>fasp_amg_interp_trunc</code>	273
9.44 <code>interpolation_em.c</code> File Reference	274
9.44.1 Detailed Description	275
9.44.2 Function Documentation	275
9.44.2.1 <code>fasp_amg_interp_em</code>	275
9.45 <code>io.c</code> File Reference	275
9.45.1 Detailed Description	278
9.45.2 Function Documentation	278
9.45.2.1 <code>fasp_dbsr_print</code>	278
9.45.2.2 <code>fasp_dbsr_read</code>	278
9.45.2.3 <code>fasp_dbsr_write</code>	279
9.45.2.4 <code>fasp_dbsr_write_coo</code>	280
9.45.2.5 <code>fasp_dcoo1_read</code>	281
9.45.2.6 <code>fasp_dcoo_print</code>	282
9.45.2.7 <code>fasp_dcoo_read</code>	283
9.45.2.8 <code>fasp_dcoo_shift_read</code>	284
9.45.2.9 <code>fasp_dcoo_write</code>	285
9.45.2.10 <code>fasp_dcsr_print</code>	286
9.45.2.11 <code>fasp_dcsr_read</code>	286
9.45.2.12 <code>fasp_dcsr_write_coo</code>	287
9.45.2.13 <code>fasp_dCSRvec1_read</code>	287
9.45.2.14 <code>fasp_dCSRvec1_write</code>	288
9.45.2.15 <code>fasp_dCSRvec2_read</code>	289

9.45.2.16 <code>fasp_dcsrvec2_write</code>	290
9.45.2.17 <code>fasp_dmtx_read</code>	291
9.45.2.18 <code>fasp_dmtxsym_read</code>	292
9.45.2.19 <code>fasp_dstr_print</code>	293
9.45.2.20 <code>fasp_dstr_read</code>	293
9.45.2.21 <code>fasp_dstr_write</code>	294
9.45.2.22 <code>fasp_dvec_print</code>	295
9.45.2.23 <code>fasp_dvec_read</code>	295
9.45.2.24 <code>fasp_dvec_write</code>	296
9.45.2.25 <code>fasp_dvecind_read</code>	297
9.45.2.26 <code>fasp_dvecind_write</code>	298
9.45.2.27 <code>fasp_hb_read</code>	299
9.45.2.28 <code>fasp_ivec_print</code>	300
9.45.2.29 <code>fasp_ivec_read</code>	300
9.45.2.30 <code>fasp_ivec_write</code>	301
9.45.2.31 <code>fasp_ivectind_read</code>	302
9.45.2.32 <code>fasp_matrix_read</code>	303
9.45.2.33 <code>fasp_matrix_read_bin</code>	304
9.45.2.34 <code>fasp_matrix_write</code>	305
9.45.2.35 <code>fasp_vector_read</code>	306
9.45.2.36 <code>fasp_vector_write</code>	307
9.45.3 Variable Documentation	308
9.45.3.1 <code>dlength</code>	308
9.45.3.2 <code>ilength</code>	308
9.46 <code>itsolver_bcsr.c</code> File Reference	308
9.46.1 Detailed Description	309
9.46.2 Function Documentation	309
9.46.2.1 <code>fasp_solver_bdcsr_itsolver</code>	309
9.46.2.2 <code>fasp_solver_bdcsr_krylov</code>	311
9.46.2.3 <code>fasp_solver_bdcsr_krylov_block_3</code>	312
9.46.2.4 <code>fasp_solver_bdcsr_krylov_block_4</code>	314
9.46.2.5 <code>fasp_solver_bdcsr_krylov_sweeping</code>	316
9.47 <code>itsolver_bsr.c</code> File Reference	317
9.47.1 Detailed Description	318
9.47.2 Function Documentation	318
9.47.2.1 <code>fasp_solver_dbsr_itsolver</code>	318
9.47.2.2 <code>fasp_solver_dbsr_krylov</code>	320

9.47.2.3 <code>fasp_solver_dbsr_krylov_amg</code>	321
9.47.2.4 <code>fasp_solver_dbsr_krylov_amg_nk</code>	322
9.47.2.5 <code>fasp_solver_dbsr_krylov_diag</code>	323
9.47.2.6 <code>fasp_solver_dbsr_krylov_ilu</code>	325
9.47.2.7 <code>fasp_solver_dbsr_krylov_nk_amg</code>	326
9.48 <code>itsolver_csr.c</code> File Reference	328
9.48.1 Detailed Description	329
9.48.2 Function Documentation	329
9.48.2.1 <code>fasp_solver_dcsr_itsolver</code>	329
9.48.2.2 <code>fasp_solver_dcsr_krylov</code>	330
9.48.2.3 <code>fasp_solver_dcsr_krylov_amg</code>	331
9.48.2.4 <code>fasp_solver_dcsr_krylov_amg_nk</code>	333
9.48.2.5 <code>fasp_solver_dcsr_krylov_diag</code>	335
9.48.2.6 <code>fasp_solver_dcsr_krylov_ilu</code>	336
9.48.2.7 <code>fasp_solver_dcsr_krylov_ilu_M</code>	338
9.48.2.8 <code>fasp_solver_dcsr_krylov_Schwarz</code>	340
9.49 <code>itsolver_mf.c</code> File Reference	341
9.49.1 Detailed Description	342
9.49.2 Function Documentation	342
9.49.2.1 <code>fasp_solver_itsolver</code>	342
9.49.2.2 <code>fasp_solver_itsolver_init</code>	344
9.49.2.3 <code>fasp_solver_krylov</code>	344
9.50 <code>itsolver_str.c</code> File Reference	345
9.50.1 Detailed Description	346
9.50.2 Function Documentation	346
9.50.2.1 <code>fasp_solver_dstr_itsolver</code>	346
9.50.2.2 <code>fasp_solver_dstr_krylov</code>	348
9.50.2.3 <code>fasp_solver_dstr_krylov_blockgs</code>	349
9.50.2.4 <code>fasp_solver_dstr_krylov_diag</code>	350
9.50.2.5 <code>fasp_solver_dstr_krylov_ilu</code>	351
9.51 <code>lu.c</code> File Reference	353
9.51.1 Detailed Description	354
9.51.2 Function Documentation	354
9.51.2.1 <code>fasp_smat_lu_decomp</code>	354
9.51.2.2 <code>fasp_smat_lu_solve</code>	355
9.52 <code>memory.c</code> File Reference	356
9.52.1 Detailed Description	357

9.52.2 Function Documentation	357
9.52.2.1 fasp_mem_calloc	357
9.52.2.2 fasp_mem_check	357
9.52.2.3 fasp_mem_dcsr_check	358
9.52.2.4 fasp_mem_free	358
9.52.2.5 fasp_mem_iludata_check	358
9.52.2.6 fasp_mem_realloc	359
9.52.2.7 fasp_mem_usage	359
9.52.3 Variable Documentation	360
9.52.3.1 total_alloc_count	360
9.52.3.2 total_alloc_mem	360
9.53 message.c File Reference	360
9.53.1 Detailed Description	361
9.53.2 Function Documentation	361
9.53.2.1 fasp_chkerr	361
9.53.2.2 print_amgcomplexity	361
9.53.2.3 print_amgcomplexity_bsr	362
9.53.2.4 print_cputime	362
9.53.2.5 print_itinfo	362
9.53.2.6 print_message	363
9.54 mgcycle.c File Reference	363
9.54.1 Detailed Description	364
9.54.2 Function Documentation	364
9.54.2.1 fasp_solver_mgcycle	364
9.54.2.2 fasp_solver_mgcycle_bsr	365
9.55 mgrecur.c File Reference	366
9.55.1 Detailed Description	367
9.55.2 Function Documentation	367
9.55.2.1 fasp_solver_mgrecur	367
9.56 ordering.c File Reference	368
9.56.1 Detailed Description	369
9.56.2 Function Documentation	370
9.56.2.1 fasp_aux_dQuickSort	370
9.56.2.2 fasp_aux_dQuickSortIndex	370
9.56.2.3 fasp_aux_iQuickSort	370
9.56.2.4 fasp_aux_iQuickSortIndex	371
9.56.2.5 fasp_aux_merge	371

9.56.2.6 fasp_aux_msort	372
9.56.2.7 fasp_aux_unique	373
9.56.2.8 fasp_BinarySearch	373
9.56.2.9 fasp_dcsr_CMK_order	373
9.56.2.10 fasp_dcsr_RCMK_order	374
9.57 parameters.c File Reference	374
9.57.1 Detailed Description	376
9.57.2 Function Documentation	376
9.57.2.1 fasp_param_amg_init	376
9.57.2.2 fasp_param_amg_print	376
9.57.2.3 fasp_param_amg_set	377
9.57.2.4 fasp_param_amg_to_prec	377
9.57.2.5 fasp_param_amg_to_prec_bsr	377
9.57.2.6 fasp_param_ilu_init	378
9.57.2.7 fasp_param_ilu_print	378
9.57.2.8 fasp_param_ilu_set	378
9.57.2.9 fasp_param_init	379
9.57.2.10 fasp_param_input_init	380
9.57.2.11 fasp_param_prec_to_amg	381
9.57.2.12 fasp_param_prec_to_amg_bsr	381
9.57.2.13 fasp_param_Schwarz_init	381
9.57.2.14 fasp_param_Schwarz_print	382
9.57.2.15 fasp_param_Schwarz_set	382
9.57.2.16 fasp_param_set	382
9.57.2.17 fasp_param_solver_init	383
9.57.2.18 fasp_param_solver_print	383
9.57.2.19 fasp_param_solver_set	384
9.58 pbcgs.c File Reference	384
9.58.1 Detailed Description	385
9.58.2 Function Documentation	386
9.58.2.1 fasp_solver_bdcsr_pbcgs	386
9.58.2.2 fasp_solver_dbsr_pbcgs	387
9.58.2.3 fasp_solver_dcsr_pbcgs	389
9.58.2.4 fasp_solver_dstr_pbcgs	391
9.59 pbcgs_mf.c File Reference	392
9.59.1 Detailed Description	393
9.59.2 Function Documentation	394

9.59.2.1 fasp_solver_pbcgs	394
9.60 pcg.c File Reference	395
9.60.1 Detailed Description	396
9.60.2 Function Documentation	397
9.60.2.1 fasp_solver_bdcsr_pcg	397
9.60.2.2 fasp_solver_dbsr_pcg	398
9.60.2.3 fasp_solver_dcsr_pcg	400
9.60.2.4 fasp_solver_dstr_pcg	401
9.61 pcg_mf.c File Reference	403
9.61.1 Detailed Description	403
9.61.2 Function Documentation	404
9.61.2.1 fasp_solver_pcg	404
9.62 pgcg.c File Reference	406
9.62.1 Detailed Description	406
9.62.2 Function Documentation	406
9.62.2.1 fasp_solver_dcsr_pgcg	406
9.63 pgcg_mf.c File Reference	408
9.63.1 Detailed Description	408
9.63.2 Function Documentation	408
9.63.2.1 fasp_solver_pgcg	408
9.64 pgcr.c File Reference	410
9.64.1 Detailed Description	410
9.64.2 Function Documentation	410
9.64.2.1 fasp_solver_dcsr_pgcr	410
9.64.2.2 fasp_solver_dcsr_pgcr1	411
9.65 pgmres.c File Reference	413
9.65.1 Detailed Description	414
9.65.2 Function Documentation	414
9.65.2.1 fasp_solver_bdcsr_pgmres	414
9.65.2.2 fasp_solver_dbsr_pgmres	415
9.65.2.3 fasp_solver_dcsr_pgmres	416
9.65.2.4 fasp_solver_dstr_pgmres	418
9.66 pgmres_mf.c File Reference	419
9.66.1 Detailed Description	420
9.66.2 Function Documentation	420
9.66.2.1 fasp_solver_pgmres	420
9.67 pminres.c File Reference	421

9.67.1 Detailed Description	422
9.67.2 Function Documentation	423
9.67.2.1 fasp_solver_bdcsr_pminres	423
9.67.2.2 fasp_solver_dcsr_pminres	425
9.67.2.3 fasp_solver_dstr_pminres	427
9.68 pminres_mf.c File Reference	428
9.68.1 Detailed Description	429
9.68.2 Function Documentation	430
9.68.2.1 fasp_solver_pminres	430
9.69 precond_bcsr.c File Reference	431
9.69.1 Detailed Description	432
9.69.2 Function Documentation	433
9.69.2.1 fasp_precond_block_diag_3	433
9.69.2.2 fasp_precond_block_diag_3_amg	434
9.69.2.3 fasp_precond_block_diag_4	435
9.69.2.4 fasp_precond_block_lower_3	436
9.69.2.5 fasp_precond_block_lower_3_amg	437
9.69.2.6 fasp_precond_block_lower_4	438
9.69.2.7 fasp_precond_sweeping	439
9.70 precond_bsr.c File Reference	440
9.70.1 Detailed Description	441
9.70.2 Function Documentation	441
9.70.2.1 fasp_precond_dbsr_amg	441
9.70.2.2 fasp_precond_dbsr_amg_nk	442
9.70.2.3 fasp_precond_dbsr_diag	443
9.70.2.4 fasp_precond_dbsr_diag_nc2	444
9.70.2.5 fasp_precond_dbsr_diag_nc3	445
9.70.2.6 fasp_precond_dbsr_diag_nc5	446
9.70.2.7 fasp_precond_dbsr_diag_nc7	447
9.70.2.8 fasp_precond_dbsr_ilu	448
9.70.2.9 fasp_precond_dbsr_nl_amli	449
9.71 precond_csr.c File Reference	450
9.71.1 Detailed Description	451
9.71.2 Function Documentation	452
9.71.2.1 fasp_precond_amg	452
9.71.2.2 fasp_precond_amg_nk	452
9.71.2.3 fasp_precond_amli	453

9.71.2.4 fasp_precond_diag	454
9.71.2.5 fasp_precond_famg	455
9.71.2.6 fasp_precond_free	456
9.71.2.7 fasp_precond_ilu	457
9.71.2.8 fasp_precond_ilu_backward	458
9.71.2.9 fasp_precond_ilu_forward	458
9.71.2.10 fasp_precond_nl_amli	459
9.71.2.11 fasp_precond_Schwarz	460
9.71.2.12 fasp_precond_setup	460
9.72 precond_str.c File Reference	462
9.72.1 Detailed Description	462
9.72.2 Function Documentation	463
9.72.2.1 fasp_precond_dstr_blockgs	463
9.72.2.2 fasp_precond_dstr_diag	463
9.72.2.3 fasp_precond_dstr_ilu0	464
9.72.2.4 fasp_precond_dstr_ilu0_backward	465
9.72.2.5 fasp_precond_dstr_ilu0_forward	466
9.72.2.6 fasp_precond_dstr_ilu1	467
9.72.2.7 fasp_precond_dstr_ilu1_backward	468
9.72.2.8 fasp_precond_dstr_ilu1_forward	469
9.73 pvfgmres.c File Reference	470
9.73.1 Detailed Description	471
9.73.2 Function Documentation	471
9.73.2.1 fasp_solver_bdcsr_pvfgmres	472
9.73.2.2 fasp_solver_dbsr_pvfgmres	474
9.73.2.3 fasp_solver_dcsr_pvfgmres	475
9.74 pvfgmres_mf.c File Reference	477
9.74.1 Detailed Description	477
9.74.2 Function Documentation	477
9.74.2.1 fasp_solver_pvfgmres	477
9.75 pvgmres.c File Reference	479
9.75.1 Detailed Description	479
9.75.2 Function Documentation	480
9.75.2.1 fasp_solver_bdcsr_pvgmres	480
9.75.2.2 fasp_solver_dbsr_pvgmres	481
9.75.2.3 fasp_solver_dcsr_pvgmres	482
9.75.2.4 fasp_solver_dstr_pvgmres	484

9.76 pvgmres_mf.c File Reference	485
9.76.1 Detailed Description	486
9.76.2 Function Documentation	486
9.76.2.1 fasp_solver_pvgmres	486
9.77 quadrature.c File Reference	487
9.77.1 Detailed Description	488
9.77.2 Function Documentation	488
9.77.2.1 fasp_gauss2d	488
9.77.2.2 fasp_quad2d	489
9.78 rap.c File Reference	490
9.78.1 Detailed Description	490
9.78.2 Function Documentation	490
9.78.2.1 fasp blas dcsr rap2	490
9.79 schwarz.f File Reference	491
9.79.1 Detailed Description	492
9.80 schwarz_setup.c File Reference	492
9.80.1 Detailed Description	493
9.80.2 Function Documentation	493
9.80.2.1 fasp_dcsr_Schwarz_backward_smoothen	493
9.80.2.2 fasp_dcsr_Schwarz_forward_smoothen	494
9.80.2.3 fasp_Schwarz_get_block_matrix	495
9.80.2.4 fasp_Schwarz_setup	496
9.81 smat.c File Reference	497
9.81.1 Detailed Description	499
9.81.2 Function Documentation	499
9.81.2.1 fasp blas smat inv	499
9.81.2.2 fasp blas smat inv nc2	499
9.81.2.3 fasp blas smat inv nc3	500
9.81.2.4 fasp blas smat inv nc4	500
9.81.2.5 fasp blas smat inv nc5	500
9.81.2.6 fasp blas smat inv nc7	501
9.81.2.7 fasp blas smat Linfinity	501
9.81.2.8 fasp iden free	502
9.81.2.9 fasp smat identity	502
9.81.2.10 fasp smat identity nc2	503
9.81.2.11 fasp smat identity nc3	503
9.81.2.12 fasp smat identity nc5	503

9.81.2.13 fasp_smat_identity_nc7	504
9.82 smoother_bsr.c File Reference	504
9.82.1 Detailed Description	506
9.82.2 Function Documentation	506
9.82.2.1 fasp_smoothen_dbsr_gs	506
9.82.2.2 fasp_smoothen_dbsr_gs1	507
9.82.2.3 fasp_smoothen_dbsr_gs_ascend	508
9.82.2.4 fasp_smoothen_dbsr_gs_ascend1	509
9.82.2.5 fasp_smoothen_dbsr_gs_descend	510
9.82.2.6 fasp_smoothen_dbsr_gs_descend1	511
9.82.2.7 fasp_smoothen_dbsr_gs_order1	512
9.82.2.8 fasp_smoothen_dbsr_gs_order2	513
9.82.2.9 fasp_smoothen_dbsr_ilu	514
9.82.2.10 fasp_smoothen_dbsr_jacobi	515
9.82.2.11 fasp_smoothen_dbsr_jacobi1	516
9.82.2.12 fasp_smoothen_dbsr_jacobi_setup	517
9.82.2.13 fasp_smoothen_dbsr_sor	518
9.82.2.14 fasp_smoothen_dbsr_sor1	519
9.82.2.15 fasp_smoothen_dbsr_sor_ascend	520
9.82.2.16 fasp_smoothen_dbsr_sor_descend	521
9.82.2.17 fasp_smoothen_dbsr_sor_order	522
9.83 smoother_csr.c File Reference	523
9.83.1 Detailed Description	525
9.83.2 Function Documentation	525
9.83.2.1 fasp_smoothen_dcsr_gs	525
9.83.2.2 fasp_smoothen_dcsr_gs_cf	525
9.83.2.3 fasp_smoothen_dcsr_gs_rb3d	526
9.83.2.4 fasp_smoothen_dcsr_ilu	527
9.83.2.5 fasp_smoothen_dcsr_jacobi	528
9.83.2.6 fasp_smoothen_dcsr_kaczmarz	529
9.83.2.7 fasp_smoothen_dcsr_L1diag	530
9.83.2.8 fasp_smoothen_dcsr_sgs	531
9.83.2.9 fasp_smoothen_dcsr_sor	532
9.83.2.10 fasp_smoothen_dcsr_sor_cf	533
9.84 smoother_csr_cr.c File Reference	534
9.84.1 Detailed Description	535
9.84.2 Function Documentation	535

9.84.2.1 fasp_smoothen_dcsr_gscr	535
9.85 smoother_csr_poly.c File Reference	536
9.85.1 Detailed Description	536
9.85.2 Function Documentation	536
9.85.2.1 fasp_smoothen_dcsr_poly	536
9.85.2.2 fasp_smoothen_dcsr_poly_old	537
9.86 smoother_str.c File Reference	538
9.86.1 Detailed Description	539
9.86.2 Function Documentation	539
9.86.2.1 fasp_generate_diaginv_block	539
9.86.2.2 fasp_smoothen_dstr_gs	540
9.86.2.3 fasp_smoothen_dstr_gs1	541
9.86.2.4 fasp_smoothen_dstr_gs_ascend	542
9.86.2.5 fasp_smoothen_dstr_gs_cf	543
9.86.2.6 fasp_smoothen_dstr_gs_descend	544
9.86.2.7 fasp_smoothen_dstr_gs_order	545
9.86.2.8 fasp_smoothen_dstr_jacobi	546
9.86.2.9 fasp_smoothen_dstr_jacobi1	547
9.86.2.10 fasp_smoothen_dstr_schwarz	548
9.86.2.11 fasp_smoothen_dstr_sor	549
9.86.2.12 fasp_smoothen_dstr_sor1	550
9.86.2.13 fasp_smoothen_dstr_sor_ascend	551
9.86.2.14 fasp_smoothen_dstr_sor_cf	552
9.86.2.15 fasp_smoothen_dstr_sor_descend	553
9.86.2.16 fasp_smoothen_dstr_sor_order	554
9.87 sparse_block.c File Reference	555
9.87.1 Detailed Description	556
9.87.2 Function Documentation	556
9.87.2.1 fasp_bdcsr_free	556
9.87.2.2 fasp_dbср_getblk	556
9.87.2.3 fasp_dbср_getblk_dCSR	557
9.87.2.4 fasp_dbср_Linfinity_dCSR	558
9.87.2.5 fasp_dCSR_getblk	559
9.88 sparse_bsr.c File Reference	560
9.88.1 Detailed Description	561
9.88.2 Function Documentation	561
9.88.2.1 fasp_dbср_alloc	561

9.88.2.2 fasp_dbsr_cp	562
9.88.2.3 fasp_dbsr_create	562
9.88.2.4 fasp_dbsr_diaginv	563
9.88.2.5 fasp_dbsr_diaginv2	564
9.88.2.6 fasp_dbsr_diaginv3	565
9.88.2.7 fasp_dbsr_diaginv4	566
9.88.2.8 fasp_dbsr_diagLU	567
9.88.2.9 fasp_dbsr_diagLU2	568
9.88.2.10 fasp_dbsr_diagpref	569
9.88.2.11 fasp_dbsr_free	570
9.88.2.12 fasp_dbsr_getdiag	571
9.88.2.13 fasp_dbsr_getdiaginv	571
9.88.2.14 fasp_dbsr_null	572
9.88.2.15 fasp_dbsr_trans	572
9.89 sparse_coo.c File Reference	573
9.89.1 Detailed Description	574
9.89.2 Function Documentation	574
9.89.2.1 fasp_dcoo_alloc	574
9.89.2.2 fasp_dcoo_create	574
9.89.2.3 fasp_dcoo_free	575
9.89.2.4 fasp_dcoo_shift	576
9.90 sparse_csr.c File Reference	576
9.90.1 Detailed Description	578
9.90.2 Function Documentation	578
9.90.2.1 fasp_dcsr_alloc	578
9.90.2.2 fasp_dcsr_compress	579
9.90.2.3 fasp_dcsr_compress_inplace	580
9.90.2.4 fasp_dcsr_cp	581
9.90.2.5 fasp_dcsr_create	582
9.90.2.6 fasp_dcsr_diagpref	583
9.90.2.7 fasp_dcsr_free	584
9.90.2.8 fasp_dcsr_getcol	585
9.90.2.9 fasp_dcsr_getdiag	585
9.90.2.10 fasp_dcsr_multicoloring	586
9.90.2.11 fasp_dcsr_null	586
9.90.2.12 fasp_dcsr_perm	587
9.90.2.13 fasp_dcsr_regdiag	587

9.90.2.14 <code>fasp_dcsr_shift</code>	588
9.90.2.15 <code>fasp_dcsr_sort</code>	588
9.90.2.16 <code>fasp_dcsr_symdiagscale</code>	589
9.90.2.17 <code>fasp_dcsr_sympat</code>	590
9.90.2.18 <code>fasp_dcsr_trans</code>	591
9.90.2.19 <code>fasp_icsr_cp</code>	592
9.90.2.20 <code>fasp_icsr_create</code>	593
9.90.2.21 <code>fasp_icsr_free</code>	594
9.90.2.22 <code>fasp_icsr_null</code>	594
9.90.2.23 <code>fasp_icsr_trans</code>	595
9.91 <code>sparse_csr.c</code> File Reference	595
9.91.1 Detailed Description	596
9.91.2 Function Documentation	596
9.91.2.1 <code>fasp_dcsr_create</code>	596
9.91.2.2 <code>fasp_dcsr_free</code>	597
9.92 <code>sparse_str.c</code> File Reference	597
9.92.1 Detailed Description	598
9.92.2 Function Documentation	598
9.92.2.1 <code>fasp_dstr_alloc</code>	598
9.92.2.2 <code>fasp_dstr_cp</code>	599
9.92.2.3 <code>fasp_dstr_create</code>	600
9.92.2.4 <code>fasp_dstr_free</code>	601
9.92.2.5 <code>fasp_dstr_null</code>	602
9.93 <code>sparse_util.c</code> File Reference	602
9.93.1 Detailed Description	604
9.93.2 Function Documentation	604
9.93.2.1 <code>fasp_sparse_aat</code>	604
9.93.2.2 <code>fasp_sparse_abyb</code>	604
9.93.2.3 <code>fasp_sparse_abybms</code>	605
9.93.2.4 <code>fasp_sparse_aplbms</code>	605
9.93.2.5 <code>fasp_sparse_aplusb</code>	606
9.93.2.6 <code>fasp_sparse_iit</code>	606
9.93.2.7 <code>fasp_sparse_MIS</code>	606
9.93.2.8 <code>fasp_sparse_rapcmp</code>	607
9.93.2.9 <code>fasp_sparse_raprms</code>	608
9.93.2.10 <code>fasp_sparse_wta</code>	608
9.93.2.11 <code>fasp_sparse_wtams</code>	609

9.93.2.12 fasp_sparse_ytx_	609
9.93.2.13 fasp_sparse_ytxbig_	610
9.94 spbcgs.c File Reference	610
9.94.1 Detailed Description	611
9.94.2 Function Documentation	612
9.94.2.1 fasp_solver_bdcsr_spbcgs	612
9.94.2.2 fasp_solver_dbsr_spbcgs	613
9.94.2.3 fasp_solver_dcsr_spbcgs	614
9.94.2.4 fasp_solver_dstr_spbcgs	616
9.95 spcg.c File Reference	618
9.95.1 Detailed Description	618
9.95.2 Function Documentation	619
9.95.2.1 fasp_solver_bdcsr_spcg	619
9.95.2.2 fasp_solver_dcsr_spcg	621
9.95.2.3 fasp_solver_dstr_spcg	623
9.96 spgmres.c File Reference	625
9.96.1 Detailed Description	625
9.96.2 Function Documentation	626
9.96.2.1 fasp_solver_bdcsr_spgmres	626
9.96.2.2 fasp_solver_dbsr_spgmres	627
9.96.2.3 fasp_solver_dcsr_spgmres	628
9.96.2.4 fasp_solver_dstr_spgmres	630
9.97 spminres.c File Reference	632
9.97.1 Detailed Description	632
9.97.2 Function Documentation	634
9.97.2.1 fasp_solver_bdcsr_spminres	634
9.97.2.2 fasp_solver_dcsr_spminres	635
9.97.2.3 fasp_solver_dstr_spminres	637
9.98 spvgmres.c File Reference	638
9.98.1 Detailed Description	639
9.98.2 Function Documentation	639
9.98.2.1 fasp_solver_bdcsr_spvgmres	640
9.98.2.2 fasp_solver_dbsr_spvgmres	641
9.98.2.3 fasp_solver_dcsr_spvgmres	642
9.98.2.4 fasp_solver_dstr_spvgmres	644
9.99 threads.c File Reference	646
9.99.1 Detailed Description	646

9.99.2 Function Documentation	646
9.99.2.1 FASP_GET_START_END	646
9.99.2.2 fasp_set_GS_threads	647
9.99.3 Variable Documentation	647
9.99.3.1 THDs_AMG_GS	647
9.99.3.2 THDs_CPR_gGS	647
9.99.3.3 THDs_CPR_IGS	648
9.100 timing.c File Reference	648
9.100.1 Detailed Description	648
9.100.2 Function Documentation	648
9.100.2.1 fasp_gettime	648
9.101 vec.c File Reference	649
9.101.1 Detailed Description	650
9.101.2 Function Documentation	650
9.101.2.1 fasp_dvec_alloc	650
9.101.2.2 fasp_dvec_cp	651
9.101.2.3 fasp_dvec_create	651
9.101.2.4 fasp_dvec_free	652
9.101.2.5 fasp_dvec_isnan	652
9.101.2.6 fasp_dvec_maxdiff	653
9.101.2.7 fasp_dvec_null	654
9.101.2.8 fasp_dvec_rand	654
9.101.2.9 fasp_dvec_set	654
9.101.2.10 fasp_dvec_symdiagscale	655
9.101.2.11 fasp_ivector_alloc	656
9.101.2.12 fasp_ivector_create	656
9.101.2.13 fasp_ivector_free	657
9.101.2.14 fasp_ivector_set	658
9.102 wrapper.c File Reference	659
9.102.1 Detailed Description	659
9.102.2 Function Documentation	660
9.102.2.1 fasp_fwrapper_amg	660
9.102.2.2 fasp_fwrapper_krylov_amg	661
9.102.2.3 fasp_wrapper_dbsr_krylov_amg	662
9.102.2.4 fasp_wrapper_dcoo_dbsr_krylov_amg	663
Index	665

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 systems from 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 P \leftarrow DE 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

For the moment, FASP is still under alpha testing. You need a password to download the package. Sorry about it!

The most updated version of FASP can be downloaded from

<http://fasp.sourceforge.net/download/faspSolver.zip>

We use HG (Mercurial) 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)

Active Developers (in alphabetic order):

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

With contributions from (in alphabetic order):

- Brannick, James (Penn State University, USA)
- Cao, Fei (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)

- 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)

Project Coordinator:

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

Chapter 5

Doxxygen

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

Data Structure Index

6.1 Data Structures

Here are the data structures with brief descriptions:

AMG_data	Data for AMG solvers	19
AMG_data_bsr	Data for multigrid levels. (BSR format)	20
AMG_param	Parameters for AMG solver	22
block_BSR	Block REAL matrix format for reservoir simulation	25
block_dCSRmat	Block REAL CSR matrix format	25
block_dvector	Block REAL vector structure	26
block_iCSRmat	Block INT CSR matrix format	27
block_ivector	Block INT vector structure	28
block_Reservoir	Block REAL matrix format for reservoir simulation	29
dBSRmat	Block sparse row storage matrix of REAL type	30
dCOOmat	Sparse matrix of REAL type in COO (or IJ) format	32
dCSRLmat	Sparse matrix of REAL type in CSRL format	32
dCSRmat	Sparse matrix of REAL type in CSR format	33
ddenmat	Dense matrix of REAL type	34
dSTRmat	Structure matrix of REAL type	34
dvector	Vector with n entries of REAL type	35
grid2d	Two dimensional grid data structure	36

iCOOmat	Sparse matrix of INT type in COO (or IJ) format	38
iCSRmat	Sparse matrix of INT type in CSR format	39
idenmat	Dense matrix of INT type	39
ILU_data	Data for ILU setup	40
ILU_param	Parameters for ILU	40
input_param	Input parameters	41
itsolver_param	Parameters passed to iterative solvers	48
ivector	Vector with n entries of INT type	50
Link	Struct for Links	50
linked_list	A linked list node	51
Mumps_data	Parameters for MUMPS interface	51
mxv_matfree	Matrix-vector multiplication, replace the actual matrix	52
precond	Preconditioner data and action	52
precond_block_data	Data passed to the preconditioner for block preconditioning for <code>block_dCSRmat</code> format	53
precond_block_reservoir_data	Data passed to the preconditioner for preconditioning reservoir simulation problems	55
precond_data	Data passed to the preconditioners	58
precond_data_bsr	Data passed to the preconditioners	60
precond_data_str	Data passed to the preconditioner for <code>dSTRmat</code> matrices	62
precond_diagbsr	Data passed to diagonal preconditioner for <code>dBSRmat</code> matrices	64
precond_diagstr	Data passed to diagonal preconditioner for <code>dSTRmat</code> matrices	65
precond_FASP_blkoil_data	Data passed to the preconditioner for preconditioning reservoir simulation problems	66
precond_sweeping_data	Data passed to the preconditioner for sweeping preconditioning	71
Schwarz_data	Data for Schwarz methods	74
Schwarz_param	Parameters for Schwarz method	75

Chapter 7

File Index

7.1 File List

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

amg.c	AMG method as an iterative solver (main file)	77
amg_setup_cr.c	Brannick-Falgout compatible relaxation based AMG: SETUP phase	79
amg_setup_rs.c	Ruge-Stuben AMG: SETUP phase	81
amg_setup_sa.c	Smoothed aggregation AMG: SETUP phase	84
amg_setup_ua.c	Unsmoothed aggregation AMG: SETUP phase	86
amg_solve.c	Algebraic multigrid iterations: SOLVE phase	88
amlirecur.c	Abstract AMLI multilevel iteration – recursive version	94
array.c	Array operations	101
blas_array.c	BLAS operations for arrays	105
blas_bcsr.c	BLAS operations for <code>block_dCSRmat</code> matrices	112
blas_bsr.c	BLAS operations for <code>dBSRmat</code> matrices	116
blas_csr.c	BLAS operations for <code>dCSRmat</code> matrices	128
blas_CSRLmat.c	BLAS operations for <code>dCSRmat</code> matrices	140
blas_smat.c	BLAS operations for <i>small</i> dense matrix	142
blas_str.c	BLAS operations for <code>dSTRmat</code> matrices	162
blas_vec.c	BLAS operations for vectors	165
checkmat.c	Check matrix properties	171

coarsening_cr.c	Coarsening with Brannick-Falgout strategy	176
coarsening_rs.c	Coarsening with a modified Ruge-Stuben strategy	178
convert.c	Some utilities for format conversion	180
doxygen.h	Main page for Doygen documentation	184
eigen.c	Simple subroutines for compute the extreme eigenvalues	184
factor.f	LU factoraization for CSR matrix	186
famg.c	Full AMG method as an iterative solver (main file)	186
fasp.h	Main header file for FASP	188
fasp_block.h	Main header file for FASP (block matrices)	197
fasp_const.h	Definition of all kinds of messages, including error messages, solver types, etc	199
fmgcycle.c	Abstract non-recursive full multigrid cycle	219
formats.c	Matrix format conversion routines	220
givens.c	Givens transformation	229
gmg_poisson.c	GMG method as an iterative solver for Poisson Problem	230
graphics.c	Functions for graphical output	240
ilu.f	ILU routines for preconditioning adapted from SPARSEKIT	245
ilu_setup_bsr.c	Setup Incomplete LU decomposition for <code>dBSRmat</code> matrices	246
ilu_setup_csr.c	Setup of ILU decomposition for <code>dCSRmat</code> matrices	247
ilu_setup_str.c	Setup of ILU decomposition for <code>dSTRmat</code> matrices	249
init.c	Initialize important data structures	253
input.c	Read input parameters	261
interface_mumps.c	Interface to MUMPS direct solvers	263
interface_samg.c	Interface to SAMG	265
interface_superlu.c	Interface to SuperLU direct solvers	268
interface_umfpack.c	Interface to UMFPACK direct solvers	269
interpolation.c	Interpolation operators for AMG	271
interpolation_em.c	Interpolation operators for AMG based on energy-min	274

io.c	Matrix-vector input/output subroutines	275
itsolver_bcsr.c	Iterative solvers for <code>block_dCSRmat</code> matrices	308
itsolver_bsr.c	Iterative solvers for <code>dBSRmat</code> matrices	317
itsolver_csr.c	Iterative solvers for <code>dCSRmat</code> matrices	328
itsolver_mf.c	Iterative solvers with matrix-free spmv	341
itsolver_str.c	Iterative solvers for <code>dSTRmat</code> matrices	345
lu.c	LU decomposition and direct solve for dense matrix	353
memory.c	Memory allocation and deallocation	356
message.c	Output some useful messages	360
mgcycle.c	Abstract non-recursive multigrid cycle	363
mrecur.c	Abstract multigrid cycle – recursive version	366
ordering.c	A collection of ordering, merging, removing duplicated integers functions	368
parameters.c	Initialize, set, or print input data and parameters	374
pbcgs.c	Krylov subspace methods – Preconditioned BiCGstab	384
pbcgs_mf.c	Krylov subspace methods – Preconditioned BiCGstab (matrix free)	392
pcg.c	Krylov subspace methods – Preconditioned conjugate gradient	395
pcg_mf.c	Krylov subspace methods – Preconditioned conjugate gradient (matrix free)	403
pgcg.c	Krylov subspace methods – Preconditioned Generalized CG	406
pgcg_mf.c	Krylov subspace methods – Preconditioned Generalized CG (matrix free)	408
pgcr.c	Krylov subspace methods – Preconditioned GCR	410
pgmres.c	Krylov subspace methods – Right-preconditioned GMRes	413
pgmres_mf.c	Krylov subspace methods – Preconditioned GMRes (matrix free)	419
pminres.c	Krylov subspace methods – Preconditioned minimal residual	421
pminres_mf.c	Krylov subspace methods – Preconditioned minimal residual (matrix free)	428
precond_bcsr.c	Preconditioners for block CSR matrices	431
precond_bsr.c	Preconditioners for <code>dBSRmat</code> matrices	440
precond_csr.c	Preconditioners for <code>dCSRmat</code> matrices	450

precond_str.c	Preconditioners for dSTRmat matrices	462
pfgmres.c	Krylov subspace methods – Preconditioned variable-restarting flexible GMRes	470
pfgmres_mf.c	Krylov subspace methods – Preconditioned variable-restarting flexible GMRes (matrix free)	477
pvgmres.c	Krylov subspace methods – Preconditioned variable-restart GMRes	479
pvgmres_mf.c	Krylov subspace methods – Preconditioned variable-restarting GMRes (matrix free)	485
quadrature.c	Quadrature rules	487
rap.c	R*A*P driver	490
schwarz.f	Schwarz smoothers	491
schwarz_setup.c	Setup phase for the Schwarz methods	492
smat.c	Simple operations for <i>small</i> dense matrices in row-major format	497
smoother_bsr.c	Smoothers for dBSRmat matrices	504
smoother_csr.c	Smoothers for dCSRmat matrices	523
smoother_csr_cr.c	Smoothers for dCSRmat matrices using compatible relaxation	534
smoother_csr_poly.c	Smoothers for dCSRmat matrices using poly. approx. to A^{-1}	536
smoother_str.c	Smoothers for dSTRmat matrices	538
sparse_block.c	Sparse matrix block operations	555
sparse_bsr.c	Sparse matrix operations for dBSRmat matrices	560
sparse_coo.c	Sparse matrix operations for dCOOmat matrices	573
sparse_csr.c	Sparse matrix operations for dCSRmat matrices	576
sparse_csr1.c	Sparse matrix operations for dCSRLmat matrices	595
sparse_str.c	Sparse matrix operations for dSTRmat matrices	597
sparse_util.c	Routines for sparse matrix operations	602
spbcgs.c	Krylov subspace methods – Preconditioned BiCGstab with safe net	610
spcg.c	Krylov subspace methods – Preconditioned conjugate gradient with safe net	618
spgmres.c	Krylov subspace methods – Preconditioned GMRes with safe net	625
spminres.c	Krylov subspace methods – Preconditioned minimal residual with safe net	632
spvgmres.c	Krylov subspace methods – Preconditioned variable-restart GMRes with safe net	638

threads.c	Get and set number of threads and assign work load for each thread	646
timing.c	Timing subroutines	648
vec.c	Simple operations for vectors	649
wrapper.c	Wrappers for accessing functions by advanced users	659

Chapter 8

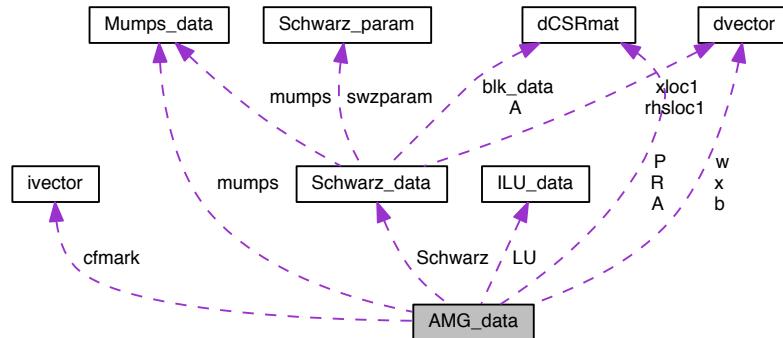
Data Structure Documentation

8.1 AMG_data Struct Reference

Data for AMG solvers.

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

Collaboration diagram for AMG_data:



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`

- **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
- **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

8.1.1 Detailed Description

Data for AMG solvers.

Note

This is needed for the AMG solver/preconditioner.

Definition at line 683 of file fasp.h.

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

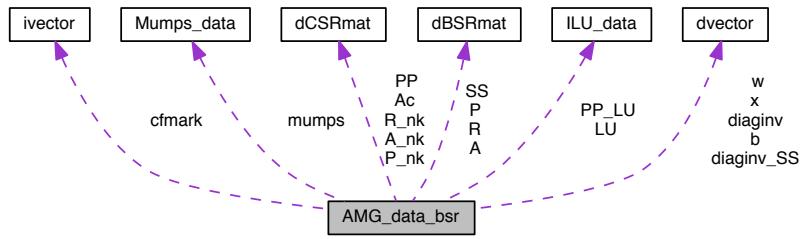
- **fasp.h**

8.2 AMG_data_bsr Struct Reference

Data for multigrid levels. (BSR format)

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

Collaboration diagram for AMG_data_bsr:



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
- `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**
Restriction for near kernal.
- **dvector w**
temporary work space
- **Mumps_data mumps**
data for MUMPS

8.2.1 Detailed Description

Data for multigrid levels. (BSR format)

Note

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

Definition at line 191 of file fasp_block.h.

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

- **fasp_block.h**

8.3 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
- **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**

- **INT pair_number**
number of paths use to determine strongly coupled C points
- **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_blk solver**
type of Schwarz block solver

8.3.1 Detailed Description

Parameters for AMG solver.

Note

This is needed for the AMG solver/preconditioner.

Definition at line 547 of file fasp.h.

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

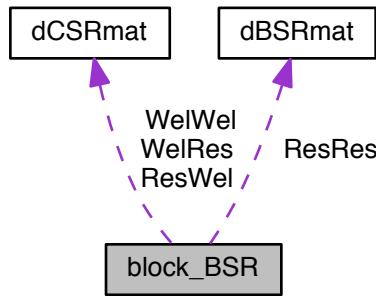
- [fasp.h](#)

8.4 block_BSR Struct Reference

Block REAL matrix format for reservoir simulation.

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

Collaboration diagram for block_BSR:



Data Fields

- `dBSRmat ResRes`
reservoir-reservoir block
- `dCSRmat ResWel`
reservoir-well block
- `dCSRmat WelRes`
well-reservoir block
- `dCSRmat WelWel`
well-well block

8.4.1 Detailed Description

Block REAL matrix format for reservoir simulation.

Definition at line 165 of file `fasp_block.h`.

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

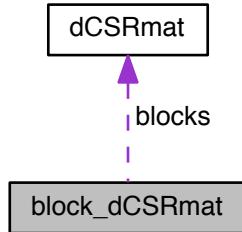
- `fasp_block.h`

8.5 block_dCSRmat Struct Reference

Block REAL CSR matrix format.

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

Collaboration diagram for block_dCSRmat:



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]

8.5.1 Detailed Description

Block REAL CSR matrix format.

Note

The starting index of A is 0.

Definition at line 77 of file `fasp_block.h`.

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

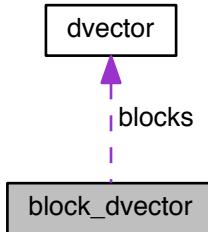
- [fasp_block.h](#)

8.6 block_dvector Struct Reference

Block REAL vector structure.

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

Collaboration diagram for block_dvector:



Data Fields

- INT brow

row number of blocks in A, m

- dvector ** blocks

blocks of dvector, point to blocks[brow]

8.6.1 Detailed Description

Block REAL vector structure.

Definition at line 113 of file fasp_block.h.

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

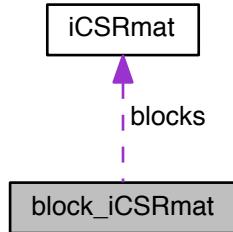
- [fasp_block.h](#)

8.7 block_iCSRmat Struct Reference

Block INT CSR matrix format.

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

Collaboration diagram for block_iCSRmat:



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]

8.7.1 Detailed Description

Block INT CSR matrix format.

Note

The starting index of A is 0.

Definition at line 96 of file `fasp_block.h`.

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

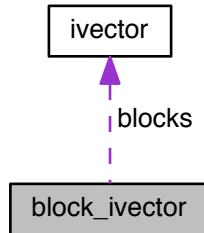
- [fasp_block.h](#)

8.8 block_ivector Struct Reference

Block INT vector structure.

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

Collaboration diagram for block_ivector:



Data Fields

- **INT brow**
row number of blocks in A, m
- **ivector ** blocks**
blocks of dvector, point to blocks[brow]

8.8.1 Detailed Description

Block INT vector structure.

Note

The starting index of A is 0.

Definition at line 129 of file fasp_block.h.

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

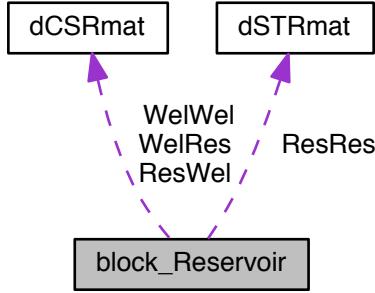
- [fasp_block.h](#)

8.9 block_Reservoir Struct Reference

Block REAL matrix format for reservoir simulation.

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

Collaboration diagram for block_Reservoir:



Data Fields

- [dSTRmat ResRes](#)
reservoir-reservoir block
- [dCSRmat ResWel](#)
reservoir-well block
- [dCSRmat WelRes](#)
well-reservoir block
- [dCSRmat WelWel](#)
well-well block

8.9.1 Detailed Description

Block REAL matrix format for reservoir simulation.

Definition at line 144 of file `fasp_block.h`.

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

- [fasp_block.h](#)

8.10 dBSPmat Struct Reference

Block sparse row storage matrix of REAL type.

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

Data Fields

- [INT ROW](#)

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

8.10.1 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 37 of file fasp_block.h.

8.10.2 Field Documentation

8.10.2.1 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 67 of file fasp_block.h.

8.10.2.2 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 60 of file fasp_block.h.

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

- [fasp_block.h](#)

8.11 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

8.11.1 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 200 of file fasp.h.

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

- **fasp.h**

8.12 dCSRLmat Struct Reference

Sparse matrix of REAL type in CSRL format.

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

Data Fields

- **INT row**
number of rows
- **INT col**

- **INT cols**
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

8.12.1 Detailed Description

Sparse matrix of REAL type in CSRL format.

Definition at line 256 of file fasp.h.

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

- **fasp.h**

8.13 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

8.13.1 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 139 of file fasp.h.

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

- [fasp.h](#)

8.14 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

8.14.1 Detailed Description

Dense matrix of REAL type.

A dense REAL matrix

Definition at line 99 of file fasp.h.

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

- [fasp.h](#)

8.15 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(nc²))*
- **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|) * nc²])*

8.15.1 Detailed Description

Structure matrix of REAL type.

Note

Every nc² 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 295 of file fasp.h.

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

- [fasp.h](#)

8.16 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

8.16.1 Detailed Description

Vector with n entries of REAL type.

Definition at line 333 of file fasp.h.

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

- [fasp.h](#)

8.17 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](#)

8.17.1 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 1085 of file fasp.h.

8.17.2 Field Documentation

8.17.2.1 INT(* e)[2]

Vertices of edges

Definition at line 1088 of file fasp.h.

8.17.2.2 INT edges

Number of edges

Definition at line 1099 of file fasp.h.

8.17.2.3 INT* ediri

Boundary flags (0 <=> interior edge)

Definition at line 1092 of file fasp.h.

8.17.2.4 INT* efather

Father edge or triangle

Definition at line 1095 of file fasp.h.

8.17.2.5 REAL(* p)[2]

Coordinates of vertices

Definition at line 1087 of file fasp.h.

8.17.2.6 INT* pdiri

Boundary flags (0 <=> interior point)

Definition at line 1091 of file fasp.h.

8.17.2.7 INT* pfather

Father point or edge

Definition at line 1094 of file fasp.h.

8.17.2.8 INT(* s)[3]

Edges of triangles

Definition at line 1090 of file fasp.h.

8.17.2.9 INT(* t)[3]

Vertices of triangles

Definition at line 1089 of file fasp.h.

8.17.2.10 INT* tfather

Father triangle

Definition at line 1096 of file fasp.h.

8.17.2.11 INT triangles

Number of triangles

Definition at line 1100 of file fasp.h.

8.17.2.12 INT vertices

Number of grid points

Definition at line 1098 of file fasp.h.

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

- [fasp.h](#)

8.18 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

8.18.1 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 230 of file fasp.h.

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

- [fasp.h](#)

8.19 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

8.19.1 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 169 of file fasp.h.

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

- [fasp.h](#)

8.20 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

8.20.1 Detailed Description

Dense matrix of INT type.

A dense INT matrix

Definition at line 118 of file fasp.h.

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

- [fasp.h](#)

8.21 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

8.21.1 Detailed Description

Data for ILU setup.

Definition at line 391 of file fasp.h.

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

- [fasp.h](#)

8.22 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)|*

8.22.1 Detailed Description

Parameters for ILU.

Definition at line 365 of file fasp.h.

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

- [fasp.h](#)

8.23 input_param Struct Reference

Input parameters.

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

Data Fields

- **SHORT print_level**
- **SHORT output_type**
- **char infile [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_smoothening
- 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_strong_coupled
- INT AMG_max_aggregation
- REAL AMG_tentative_smooth
- SHORT AMG_smooth_filter

8.23.1 Detailed Description

Input parameters.

Input parameters, reading from disk file

Definition at line 988 of file fasp.h.

8.23.2 Field Documentation

8.23.2.1 SHORT AMG_aggregation_type

aggregation type

Definition at line 1042 of file fasp.h.

8.23.2.2 INT AMG_aggressive_level

number of levels use aggressive coarsening

Definition at line 1047 of file fasp.h.

8.23.2.3 INT AMG_aggressive_path

number of paths used to determine strongly coupled C-set

Definition at line 1048 of file fasp.h.

8.23.2.4 SHORT AMG_amli_degree

degree of the polynomial used by AMLI cycle

Definition at line 1036 of file fasp.h.

8.23.2.5 INT AMG_coarse_dof

max number of coarsest level DOF

Definition at line 1030 of file fasp.h.

8.23.2.6 SHORT AMG_coarse_scaling

switch of scaling of the coarse grid correction

Definition at line 1035 of file fasp.h.

8.23.2.7 SHORT AMG_coarse_solver

coarse solver type

Definition at line 1034 of file fasp.h.

8.23.2.8 SHORT AMG_coarsening_type

coarsening type

Definition at line 1041 of file fasp.h.

8.23.2.9 SHORT AMG_cycle_type

type of cycle

Definition at line 1023 of file fasp.h.

8.23.2.10 SHORT AMG_ILU_levels

how many levels use ILU smoother

Definition at line 1033 of file fasp.h.

8.23.2.11 SHORT AMG_interpolation_type

interpolation type

Definition at line 1043 of file fasp.h.

8.23.2.12 SHORT AMG_levels

maximal number of levels

Definition at line 1022 of file fasp.h.

8.23.2.13 INT AMG_max_aggregation

max size of each aggregate

Definition at line 1053 of file fasp.h.

8.23.2.14 REAL AMG_max_row_sum

maximal row sum

Definition at line 1046 of file fasp.h.

8.23.2.15 INT AMG_maxit

number of iterations for AMG used as preconditioner

Definition at line 1032 of file fasp.h.

8.23.2.16 SHORT AMG_nl_amli_krylov_type

type of Krylov method used by nonlinear AMLI cycle

Definition at line 1037 of file fasp.h.

8.23.2.17 INT AMG_pair_number

number of pairs in matching algorithm

Definition at line 1049 of file fasp.h.

8.23.2.18 SHORT AMG_polynomial_degree

degree of the polynomial smoother

Definition at line 1027 of file fasp.h.

8.23.2.19 SHORT AMG_postsMOOTH_iter

number of postsmothing

Definition at line 1029 of file fasp.h.

8.23.2.20 SHORT AMG_presmooth_iter

number of presmoothing

Definition at line 1028 of file fasp.h.

8.23.2.21 REAL AMG_relaxation

over-relaxation parameter for SOR

Definition at line 1026 of file fasp.h.

8.23.2.22 INT AMG_Schwarz_levels

number of levels use Schwarz smoother

Definition at line 1038 of file fasp.h.

8.23.2.23 SHORT AMG_smooth_filter

use filter for smoothing the tentative prolongation or not

Definition at line 1055 of file fasp.h.

8.23.2.24 SHORT AMG_smooth_order

order for smoothers

Definition at line 1025 of file fasp.h.

8.23.2.25 SHORT AMG_smoothen

type of smoother

Definition at line 1024 of file fasp.h.

8.23.2.26 REAL AMG_strong_couple

strong coupled threshold for aggregate

Definition at line 1052 of file fasp.h.

8.23.2.27 REAL AMG_strong_threshold

strong threshold for coarsening

Definition at line 1044 of file fasp.h.

8.23.2.28 REAL AMG_tentative_smooth

relaxation factor for smoothing the tentative prolongation

Definition at line 1054 of file fasp.h.

8.23.2.29 REAL AMG_tol

tolerance for AMG if used as preconditioner

Definition at line 1031 of file fasp.h.

8.23.2.30 REAL AMG_truncation_threshold

truncation factor for interpolation

Definition at line 1045 of file fasp.h.

8.23.2.31 SHORT AMG_type

Type of AMG

Definition at line 1021 of file fasp.h.

8.23.2.32 REAL ILU_droptol

drop tolerance

Definition at line 1010 of file fasp.h.

8.23.2.33 INT ILU_Ifil

level of fill-in

Definition at line 1009 of file fasp.h.

8.23.2.34 REAL ILU_permtol

permutation tolerance

Definition at line 1012 of file fasp.h.

8.23.2.35 REAL ILU_relax

scaling factor: add the sum of dropped entries to diagonal

Definition at line 1011 of file fasp.h.

8.23.2.36 SHORT ILU_type

ILU type for decomposition

Definition at line 1008 of file fasp.h.

8.23.2.37 char infile[256]

ini file name

Definition at line 995 of file fasp.h.

8.23.2.38 INT itsolver_maxit

maximal number of iterations for iterative solvers

Definition at line 1004 of file fasp.h.

8.23.2.39 REAL itsolver_tol

tolerance for iterative linear solver

Definition at line 1003 of file fasp.h.

8.23.2.40 SHORT output_type

type of output stream

Definition at line 992 of file fasp.h.

8.23.2.41 SHORT precond_type

type of preconditioner for iterative solvers

Definition at line 1001 of file fasp.h.

8.23.2.42 SHORT print_level

print level

Definition at line 991 of file fasp.h.

8.23.2.43 INT problem_num

problem number to solve

Definition at line 997 of file fasp.h.

8.23.2.44 INT restart

restart number used in GMRES

Definition at line 1005 of file fasp.h.

8.23.2.45 INT Schwarz_blksolver

type of Schwarz block solver

Definition at line 1018 of file fasp.h.

8.23.2.46 INT Schwarz_maxlvl

maximal levels

Definition at line 1016 of file fasp.h.

8.23.2.47 INT Schwarz_mmsize

maximal block size

Definition at line 1015 of file fasp.h.

8.23.2.48 INT Schwarz_type

type of Schwarz method

Definition at line 1017 of file fasp.h.

8.23.2.49 SHORT solver_type

type of iterative solvers

Definition at line 1000 of file fasp.h.

8.23.2.50 SHORT stop_type

type of stopping criteria for iterative solvers

Definition at line 1002 of file fasp.h.

8.23.2.51 char workdir[256]

working directory for data files

Definition at line 996 of file fasp.h.

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

- [fasp.h](#)

8.24 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](#)

8.24.1 Detailed Description

Parameters passed to iterative solvers.

Definition at line 1063 of file fasp.h.

8.24.2 Field Documentation

8.24.2.1 **SHORT** itsolver_type

solver type: see message.h

Definition at line 1065 of file fasp.h.

8.24.2.2 **INT** maxit

max number of iterations

Definition at line 1068 of file fasp.h.

8.24.2.3 **SHORT** precond_type

preconditioner type: see message.h

Definition at line 1066 of file fasp.h.

8.24.2.4 **SHORT** print_level

print level: 0–10

Definition at line 1071 of file fasp.h.

8.24.2.5 **INT** restart

number of steps for restarting: for GMRES etc

Definition at line 1070 of file fasp.h.

8.24.2.6 **SHORT** stop_type

stopping criteria type

Definition at line 1067 of file fasp.h.

8.24.2.7 **REAL** tol

convergence tolerance

Definition at line 1069 of file fasp.h.

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

- [fasp.h](#)

8.25 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

8.25.1 Detailed Description

Vector with n entries of INT type.

Definition at line 347 of file fasp.h.

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

- [fasp.h](#)

8.26 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

8.26.1 Detailed Description

Struct for Links.

Definition at line 1112 of file fasp.h.

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

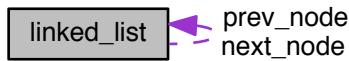
- [fasp.h](#)

8.27 linked_list Struct Reference

A linked list node.

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

Collaboration diagram for linked_list:



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

8.27.1 Detailed Description

A linked list node.

Note

This definition is adapted from hypre 2.0.

Definition at line 1129 of file fasp.h.

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

- fasp.h

8.28 Mumps_data Struct Reference

Parameters for MUMPS interface.

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

Data Fields

- **INT job**
work for MUMPS

8.28.1 Detailed Description

Parameters for MUMPS interface.

Added on 10/10/2014

Definition at line 451 of file fasp.h.

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

- [fasp.h](#)

8.29 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

8.29.1 Detailed Description

Matrix-vector multiplication, replace the actual matrix.

Definition at line 972 of file fasp.h.

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

- [fasp.h](#)

8.30 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

8.30.1 Detailed Description

Preconditioner data and action.

Note

This is the preconditioner structure for preconditioned iterative methods.

Definition at line 958 of file fasp.h.

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

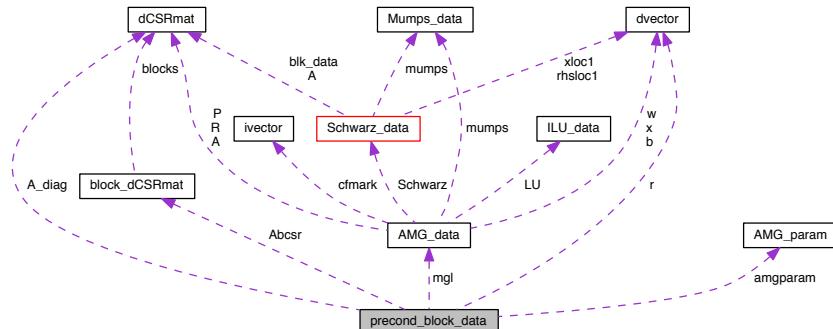
- [fasp.h](#)

8.31 precond_block_data Struct Reference

Data passed to the preconditioner for block preconditioning for [block_dCSRmat](#) format.

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

Collaboration diagram for precond_block_data:



Data Fields

- `block_dCSRmat * Abcsr`
- `dCSRmat * A_diag`
- `dvector r`
- `void ** LU_diag`
- `AMG_data ** mgli`
- `AMG_param * amgparam`

8.31.1 Detailed Description

Data passed to the preconditioner for block preconditioning for [block_dCSRmat](#) format.

Data passed to the preconditioner for block diagonal preconditioning.

This is needed for the block preconditioner.

Note

This is needed for the diagonal block preconditioner.

Definition at line 492 of file `fasp_block.h`.

8.31.2 Field Documentation

8.31.2.1 `dCSRmat* A_diag`

data for each diagonal block which need to solve in the block preconditioners

Definition at line 499 of file `fasp_block.h`.

8.31.2.2 `block_dCSRmat* Abcsr`

problem data, the blocks

Definition at line 497 of file `fasp_block.h`.

8.31.2.3 `AMG_param* amgparam`

parameters for AMG

Definition at line 511 of file `fasp_block.h`.

8.31.2.4 `void** LU_diag`

LU decomposition for the diagonal blocks – (only for UMFPack – Xiaozhe Hu)

Definition at line 507 of file `fasp_block.h`.

8.31.2.5 `AMG_data** mgl`

AMG data for the diagonal blocks

Definition at line 510 of file `fasp_block.h`.

8.31.2.6 `dvector r`

temp work space

Definition at line 501 of file `fasp_block.h`.

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

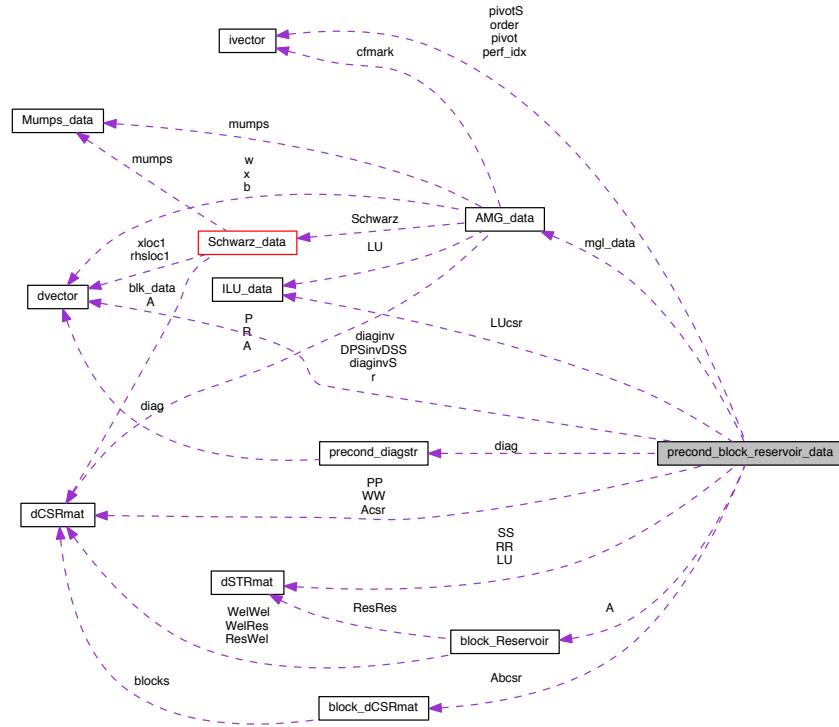
- [fasp_block.h](#)

8.32 precond_block_reservoir_data Struct Reference

Data passed to the preconditioner for preconditioning reservoir simulation problems.

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

Collaboration diagram for precond_block_reservoir_data:



Data Fields

- `block_Reservoir * A`
problem data in `block_Reservoir` format
- `block_dCSRmat * Abcsr`
problem data in `block_dCSRmat` 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 pressure-pressure 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 ($-I - A_{wr} * \text{Arr}^{-1} * A_{rw}$) $^{-1}$, Arr may be replaced by LU*
- **dvector * DPSinvDSS**
 *$\text{Diag}(PS) * \text{inv}(\text{Diag}(SS))$*
- **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**

8.32.1 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 394 of file fasp_block.h.

8.32.2 Field Documentation

8.32.2.1 `precond_diagstr* diag`

the diagonal inverse for diagonal scaling

Definition at line 474 of file fasp_block.h.

8.32.2.2 `dvector* diaginv`

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

Definition at line 475 of file fasp_block.h.

8.32.2.3 `dvector* diaginvS`

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

Definition at line 477 of file fasp_block.h.

8.32.2.4 `ivector* order`

order for smoothing

Definition at line 479 of file fasp_block.h.

8.32.2.5 `ivector* perf_idx`

variable index for perf

Definition at line 467 of file fasp_block.h.

8.32.2.6 `ivector* pivot`

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

Definition at line 476 of file fasp_block.h.

8.32.2.7 `ivector* pivotS`

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

Definition at line 478 of file fasp_block.h.

8.32.2.8 dCSRmat* PP

pressure block after diagonal scaling

Definition at line 471 of file fasp_block.h.

8.32.2.9 dvector r

temporary dvector used to store and restore the residual

Definition at line 482 of file fasp_block.h.

8.32.2.10 dSTRmat* RR

Diagonal scaled reservoir block

Definition at line 469 of file fasp_block.h.

8.32.2.11 SHORT scaled

whether the matrix is scaled

Definition at line 466 of file fasp_block.h.

8.32.2.12 dSTRmat* SS

saturation block after diaognonal scaling

Definition at line 472 of file fasp_block.h.

8.32.2.13 REAL* w

temporary work space for other usage

Definition at line 483 of file fasp_block.h.

8.32.2.14 dCSRmat* WW

Argumented well block

Definition at line 470 of file fasp_block.h.

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

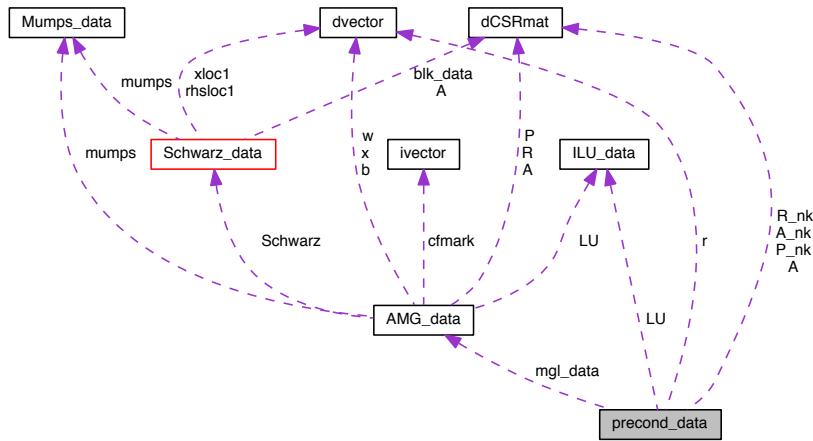
- [fasp_block.h](#)

8.33 precond_data Struct Reference

Data passed to the preconditioners.

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

Collaboration diagram for precond_data:



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

8.33.1 Detailed Description

Data passed to the preconditioners.

Definition at line 753 of file fasp.h.

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

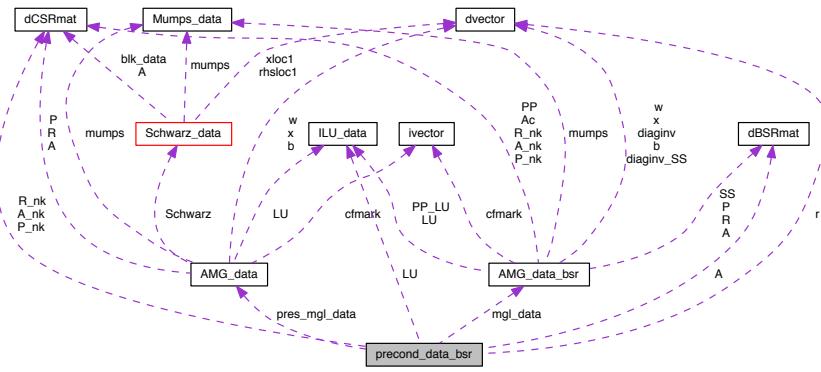
- **fasp.h**

8.34 precond_data_bsr Struct Reference

Data passed to the preconditioners.

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

Collaboration diagram for precond_data_bsr:



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**

- **REAL * aqli_coef**
degree of the polynomial used by AMLI cycle
- **REAL * aqli_coef**
coefficients of the polynomial used by AMLI cycle
- **REAL tentative_smooth**
smooth factor for smoothing the tentative prolongation
- **SHORT nl_aqli_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**
Restriction for near kernal.
- **dvector r**
temporary dvector used to store and restore the residual
- **REAL * w**
temporary work space for other usage

8.34.1 Detailed Description

Data passed to the preconditioners.

Note

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

Definition at line 301 of file fasp_block.h.

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

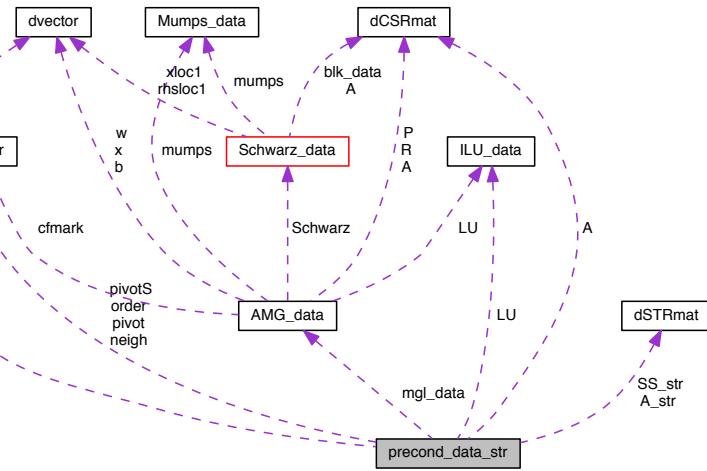
- **fasp_block.h**

8.35 precond_data_str Struct Reference

Data passed to the preconditioner for **dSTRmat** matrices.

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

Collaboration diagram for precond_data_str:



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

8.35.1 Detailed Description

Data passed to the preconditioner for **dSTRmat** matrices.

Definition at line 849 of file `fasp.h`.

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

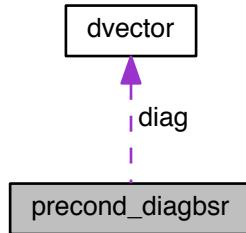
- `fasp.h`

8.36 precond_diagbsr Struct Reference

Data passed to diagonal preconditioner for **dBSRmat** matrices.

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

Collaboration diagram for precond_diagbsr:



Data Fields

- INT nb
dimension of each sub-block
- dvector diag
diagonal elements

8.36.1 Detailed Description

Data passed to diagonal preconditioner for [dBSRmat](#) matrices.

Note

This is needed for the diagonal preconditioner.

Definition at line 283 of file `fasp_block.h`.

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

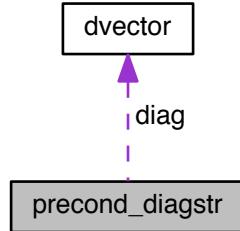
- [fasp_block.h](#)

8.37 precond_diagstr Struct Reference

Data passed to diagonal preconditioner for [dSTRmat](#) matrices.

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

Collaboration diagram for precond_diagstr:



Data Fields

- [INT nc](#)
number of components
- [dvector diag](#)
diagonal elements

8.37.1 Detailed Description

Data passed to diagonal preconditioner for [dSTRmat](#) matrices.

Note

This is needed for the diagonal preconditioner.

Definition at line 942 of file `fasp.h`.

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

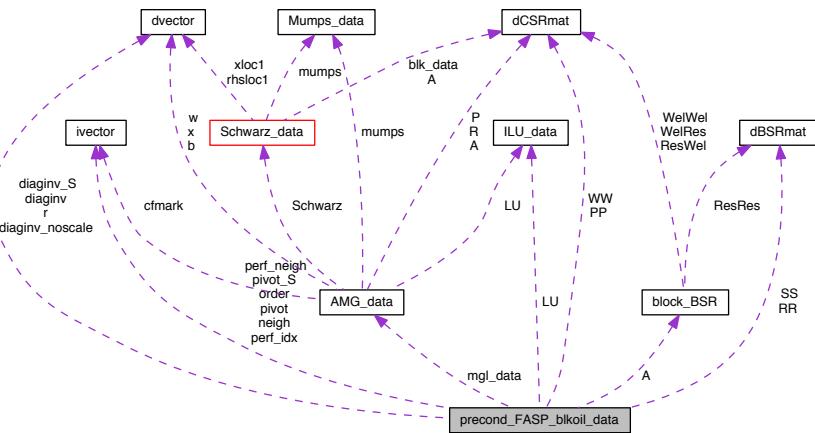
- [fasp.h](#)

8.38 precond_FASP_blkoil_data Struct Reference

Data passed to the preconditioner for preconditioning reservoir simulation problems.

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

Collaboration diagram for precond_FASP_blkoil_data:



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**
 - **dCSRmat * PP**
 - **AMG_data * mgl_data**
 - **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
- **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 $(-I - Aw^r * Arr^{-1} * Arw)^{-1}$, Arr may be replaced by LU*
- **INT maxit**
- **INT restart**
- **REAL tol**
- **dvector r**
- **REAL * w**

8.38.1 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 545 of file `fasp_block.h`.

8.38.2 Field Documentation

8.38.2.1 `block_BSR* A`

Part 1: Basic data.

whole jacobian system in `block_BSRmat`

Definition at line 550 of file `fasp_block.h`.

8.38.2.2 `dvector* diaginv`

inverse of the diagonal blocks of reservoir block

Definition at line 622 of file `fasp_block.h`.

8.38.2.3 `dvector* diaginv_noscale`

inverse of diagonal blocks for diagonal scaling

Definition at line 557 of file `fasp_block.h`.

8.38.2.4 `dvector* diaginv_S`

inverse of the diagonal blocks of saturation block

Definition at line 566 of file `fasp_block.h`.

8.38.2.5 `INT maxit`

max number of iterations

Definition at line 640 of file `fasp_block.h`.

8.38.2.6 `AMG_data* mgl_data`

AMG data for pressure-pressure block

Definition at line 571 of file `fasp_block.h`.

8.38.2.7 `ivector* neigh`

neighbor information of the reservoir block

Definition at line 561 of file `fasp_block.h`.

8.38.2.8 `ivector* order`

ordering of the reservoir block

Definition at line 562 of file `fasp_block.h`.

8.38.2.9 ivector* perf_idx

index of blocks which have perforation

Definition at line 629 of file fasp_block.h.

8.38.2.10 ivector* perf_neigh

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

Definition at line 630 of file fasp_block.h.

8.38.2.11 ivector* pivot

pivot for the GS smoothers for the reservoir matrix

Definition at line 623 of file fasp_block.h.

8.38.2.12 ivector* pivot_S

pivoting for the GS smoothers for saturation block

Definition at line 567 of file fasp_block.h.

8.38.2.13 dCSRmat* PP

pressure block

Definition at line 570 of file fasp_block.h.

8.38.2.14 dvector r

temporary dvector used to store and restore the residual

Definition at line 645 of file fasp_block.h.

8.38.2.15 INT restart

number of iterations for restart

Definition at line 641 of file fasp_block.h.

8.38.2.16 dBSPmat* RR

reservoir block

Definition at line 558 of file fasp_block.h.

8.38.2.17 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 556 of file fasp_block.h.

8.38.2.18 dBSPmat* SS

saturation block

Definition at line 565 of file fasp_block.h.

8.38.2.19 REAL tol

tolerance

Definition at line 642 of file fasp_block.h.

8.38.2.20 REAL* w

temporary work space for other usage

Definition at line 646 of file fasp_block.h.

8.38.2.21 dCSRmat* WW

Argumented well block

Definition at line 631 of file fasp_block.h.

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

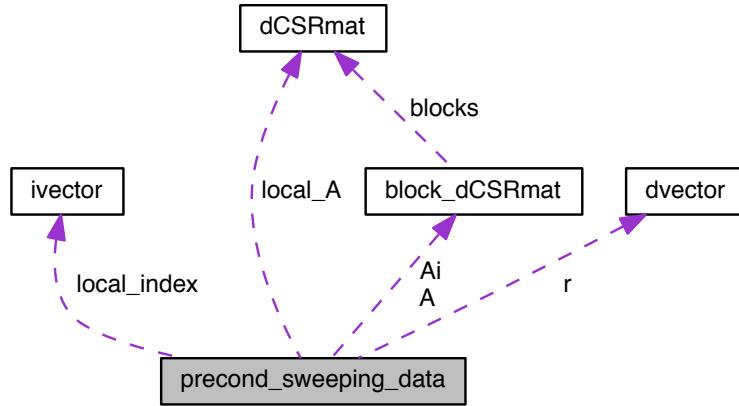
- [fasp_block.h](#)

8.39 precond_sweeping_data Struct Reference

Data passed to the preconditioner for sweeping preconditioning.

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

Collaboration diagram for precond_sweeping_data:



Data Fields

- INT NumLayers
- block_dCSRmat * A
- block_dCSRmat * Ai
- dCSRmat * local_A
- void ** local_LU
- ivecotor * local_index
- dvector r
- REAL * w

8.39.1 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 659 of file `fasp_block.h`.

8.39.2 Field Documentation

8.39.2.1 **block_dCSRmat* A**

problem data, the sparse matrix

Definition at line 663 of file fasp_block.h.

8.39.2.2 **block_dCSRmat* Ai**

preconditioner data, the sparse matrix

Definition at line 664 of file fasp_block.h.

8.39.2.3 **dCSRmat* local_A**

local stiffness matrix for each layer

Definition at line 666 of file fasp_block.h.

8.39.2.4 **ivector* local_index**

local index for each layer

Definition at line 669 of file fasp_block.h.

8.39.2.5 **void** local_LU**

Icoal LU decomposition – (only for UMFPack – Xiaozhe Hu)

Definition at line 667 of file fasp_block.h.

8.39.2.6 **INT NumLayers**

number of layers

Definition at line 661 of file fasp_block.h.

8.39.2.7 **dvector r**

temporary dvector used to store and restore the residual

Definition at line 672 of file fasp_block.h.

8.39.2.8 **REAL* w**

temporary work space for other usage

Definition at line 673 of file fasp_block.h.

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

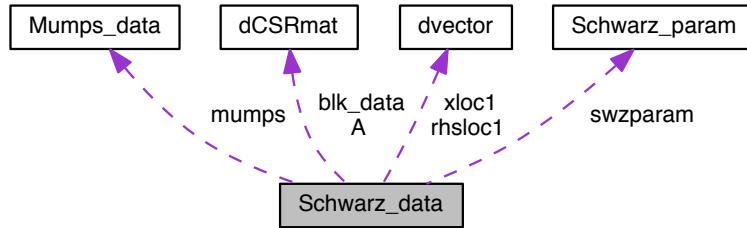
- [fasp_block.h](#)

8.40 Schwarz_data Struct Reference

Data for Schwarz methods.

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

Collaboration diagram for Schwarz_data:



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

8.40.1 Detailed Description

Data for Schwarz methods.

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

Definition at line 469 of file fasp.h.

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

- [fasp.h](#)

8.41 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

8.41.1 Detailed Description

Parameters for Schwarz method.

Added on 05/14/2012

Definition at line 426 of file fasp.h.

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

- [fasp.h](#)

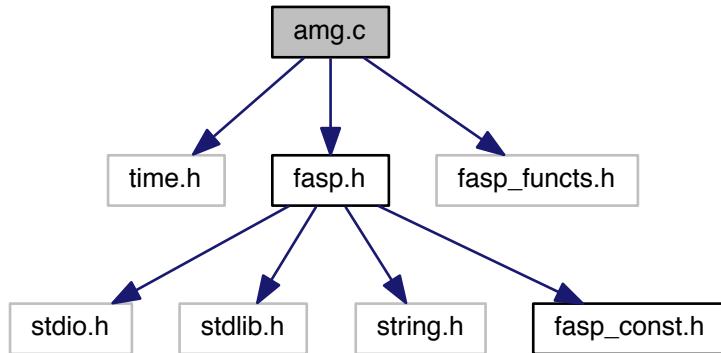
Chapter 9

File Documentation

9.1 amg.c File Reference

AMG method as an iterative solver (main file)

```
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
Include dependency graph for amg.c:
```



Functions

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

Solve $Ax = b$ by algebraic multigrid methods.

9.1.1 Detailed Description

AMG method as an iterative solver (main file)

9.1.2 Function Documentation

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

Solve $Ax = b$ by algebraic multigrid methods.

Parameters

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

Author

Chensong Zhang

Date

04/06/2010

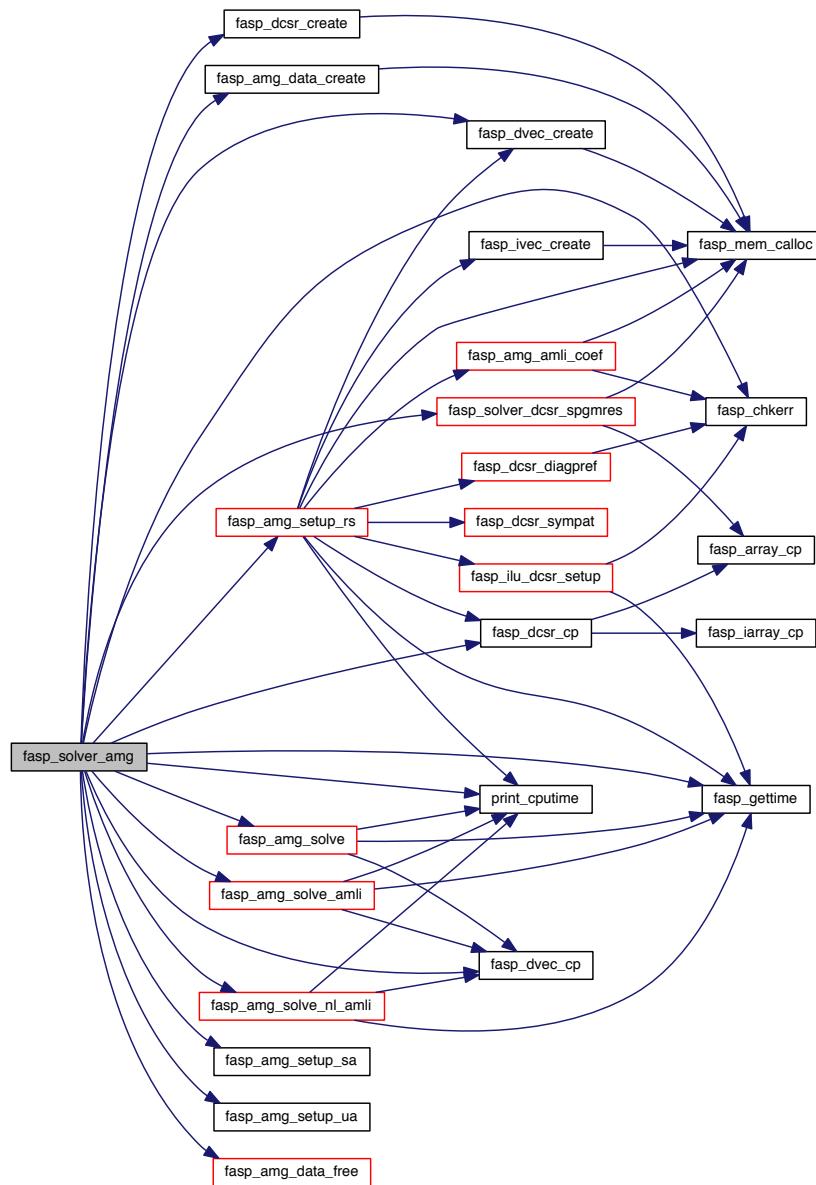
Note

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.

Here is the call graph for this function:

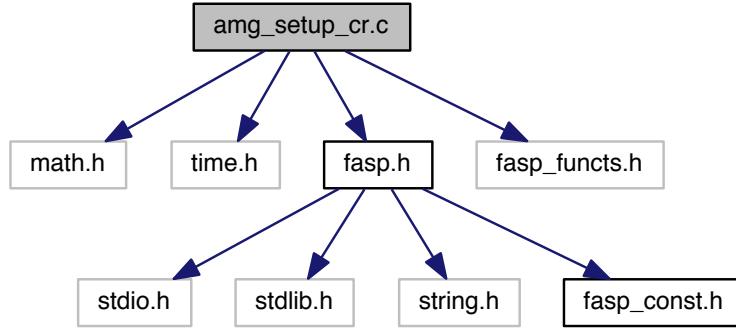


9.2 amg_setup_cr.c File Reference

Brannick-Falgout compatible relaxation based AMG: SETUP phase.

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

Include dependency graph for amg_setup_cr.c:



Functions

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

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

9.2.1 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"

TODO: Not working. Yet need to be fixed. –Chensong

9.2.2 Function Documentation

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

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

Parameters

<code>mgl</code>	Pointer to AMG data: <code>AMG_data</code>
<code>param</code>	Pointer to AMG parameters: <code>AMG_param</code>

Returns

`FASP_SUCCESS` if succeeded, otherwise, error information.

Author

James Brannick

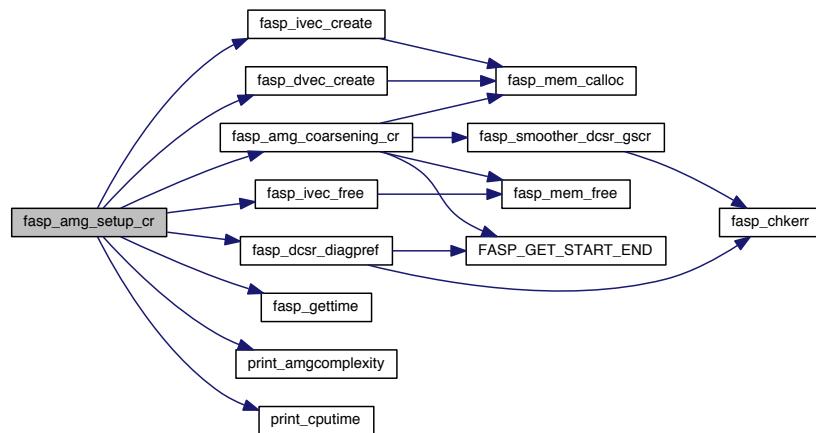
Date

04/21/2010

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

Definition at line 38 of file amg_setup_cr.c.

Here is the call graph for this function:

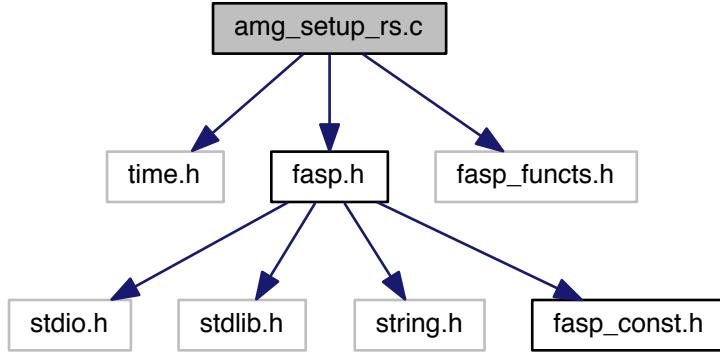


9.3 amg_setup_rs.c File Reference

Ruge-Stuben AMG: SETUP phase.

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

Include dependency graph for amg_setup_rs.c:



Functions

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

Setup phase of Ruge and Stuben's classic AMG.

9.3.1 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.

9.3.2 Function Documentation

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

Setup phase of Ruge and Stuben's classic AMG.

Parameters

<i>mgl</i>	Pointer to AMG data: AMG_data
<i>param</i>	Pointer to AMG parameters: AMG_param

Returns

FASP_SUCCESS if succeed, otherwise, error information.

Author

Chensong Zhang

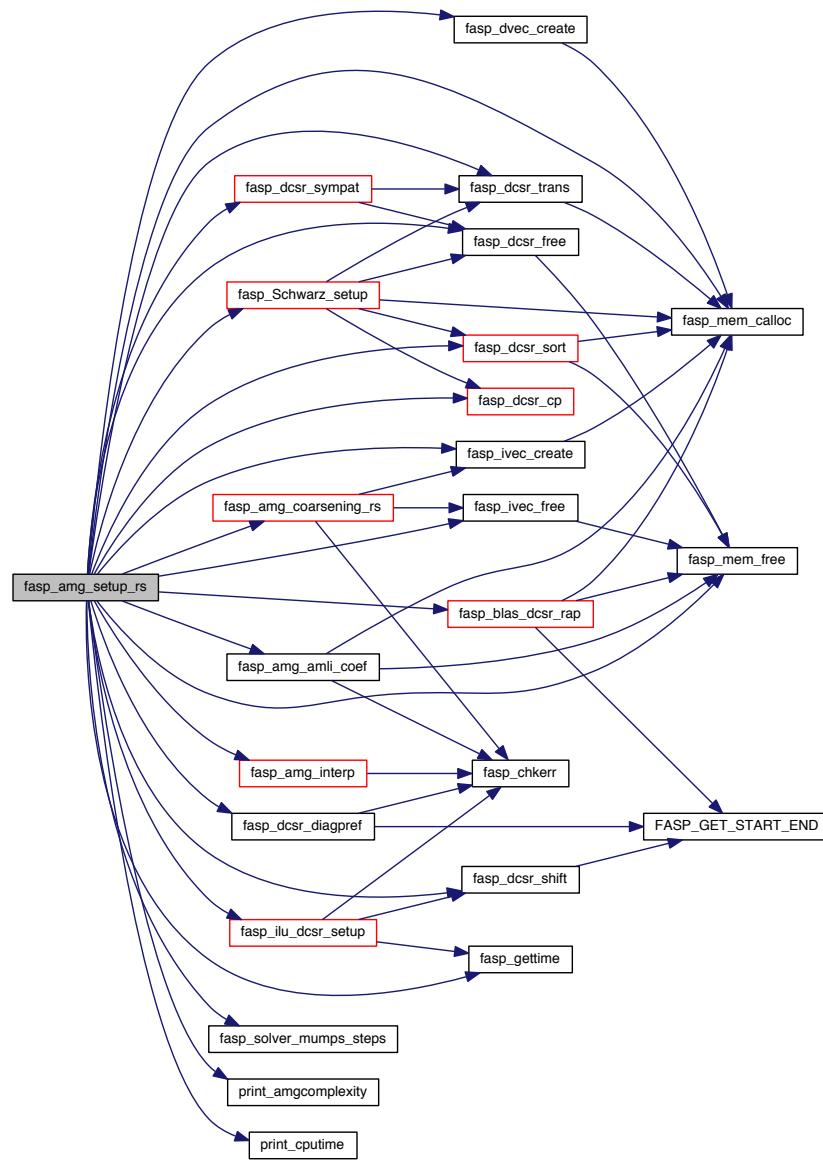
Date

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.

Here is the call graph for this function:

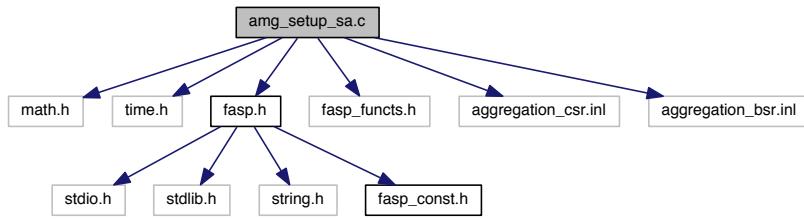


9.4 amg_setup_sa.c File Reference

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"
```

Include dependency graph for amg_setup_sa.c:



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)

9.4.1 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

9.4.2 Function Documentation

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

Set up phase of smoothed aggregation AMG.

Parameters

<i>mgl</i>	Pointer to AMG data: AMG_data
<i>param</i>	Pointer to AMG parameters: AMG_param

Returns

FASP_SUCCESS if succeed, error otherwise

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 51 of file amg_setup_sa.c.

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

Set up phase of smoothed aggregation AMG (BSR format)

Parameters

<i>mgl</i>	Pointer to AMG data: AMG_data_bsr
<i>param</i>	Pointer to AMG parameters: AMG_param

Returns

FASP_SUCCESS if succeed, error otherwise

Author

Xiaozhe Hu

Date

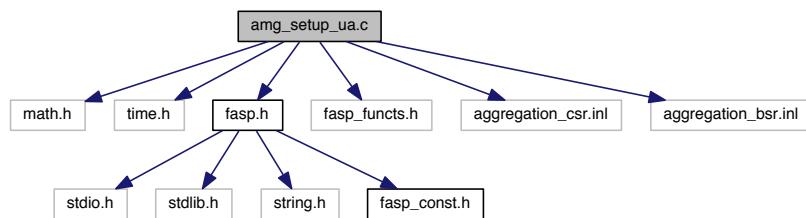
05/26/2014

Definition at line 88 of file amg_setup_sa.c.

9.5 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"
Include dependency graph for amg_setup_ua.c:
```



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)

9.5.1 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

9.5.2 Function Documentation

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

Set up phase of unsmoothed aggregation AMG.

Parameters

<i>mgl</i>	Pointer to AMG data: AMG_data
<i>param</i>	Pointer to AMG parameters: AMG_param

Returns

FASP_SUCCESS if succeed, error otherwise

Author

Xiaozhe Hu

Date

12/28/2011

Definition at line 38 of file amg_setup_ua.c.

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

Set up phase of unsmoothed aggregation AMG (BSR format)

Parameters

<i>mgl</i>	Pointer to AMG data: AMG_data_bsr
------------	---

<i>param</i>	Pointer to AMG parameters: AMG_param
--------------	--

Returns

FASP_SUCCESS if succeed, error otherwise

Author

Xiaozhe Hu

Date

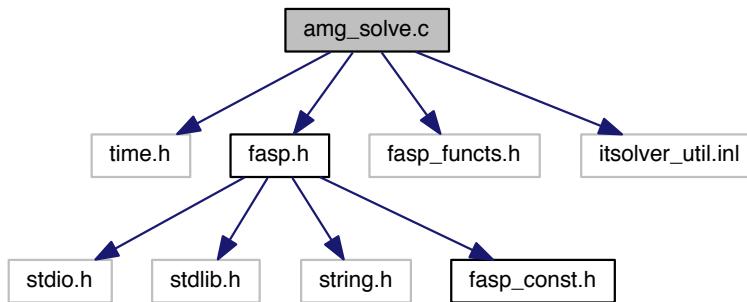
03/16/2012

Definition at line 69 of file amg_setup_ua.c.

9.6 amg_solve.c File Reference

Algebraic multigrid iterations: SOLVE phase.

```
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
Include dependency graph for amg_solve.c:
```



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.

9.6.1 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!

9.6.2 Function Documentation

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

AMG – SOLVE phase.

Parameters

<i>mgl</i>	Pointer to AMG data: AMG_data
<i>param</i>	Pointer to AMG parameters: AMG_param

Returns

Iteration number if succeed, ERROR otherwise

Author

Xuehai Huang, Chensong Zhang

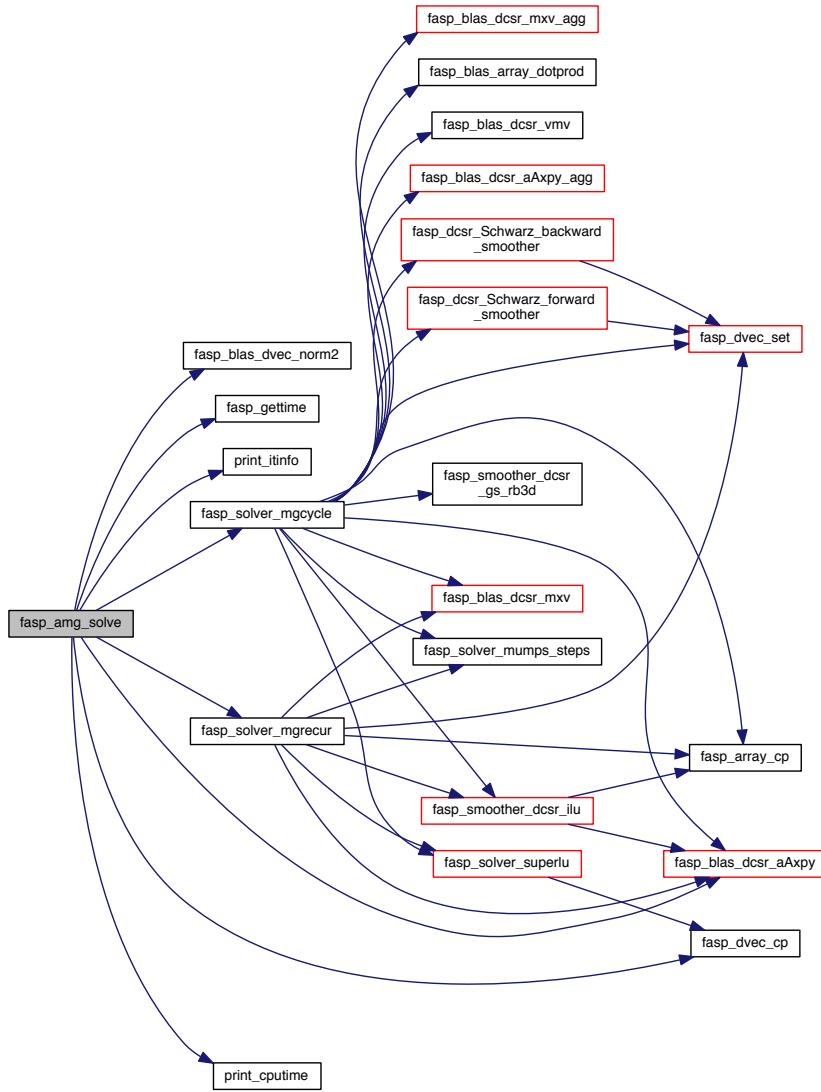
Date

04/02/2010

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

Definition at line 36 of file amg_solve.c.

Here is the call graph for this function:



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

AMLI – SOLVE phase.

Parameters

<i>mgl</i>	Pointer to AMG data: AMG_data
------------	---

<i>param</i>	Pointer to AMG parameters: AMG_param
--------------	--

Returns

Iteration number if succeed, ERROR otherwise

Author

Xiaozhe Hu

Date

01/23/2011

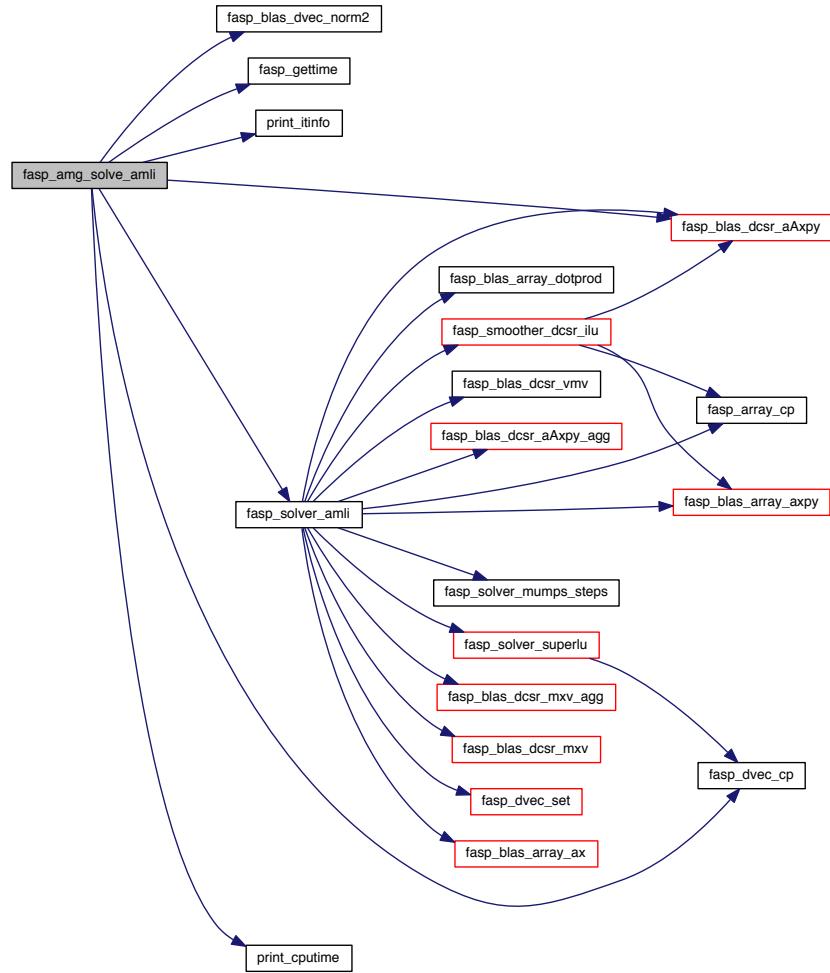
Note

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.

Here is the call graph for this function:



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

Nonlinear AMLI – SOLVE phase.

Parameters

<i>mgl</i>	Pointer to AMG data: AMG_data
<i>param</i>	Pointer to AMG parameters: AMG_param

Returns

Iteration number if succeed, ERROR otherwise

Author

Xiaozhe Hu

Date

04/30/2011

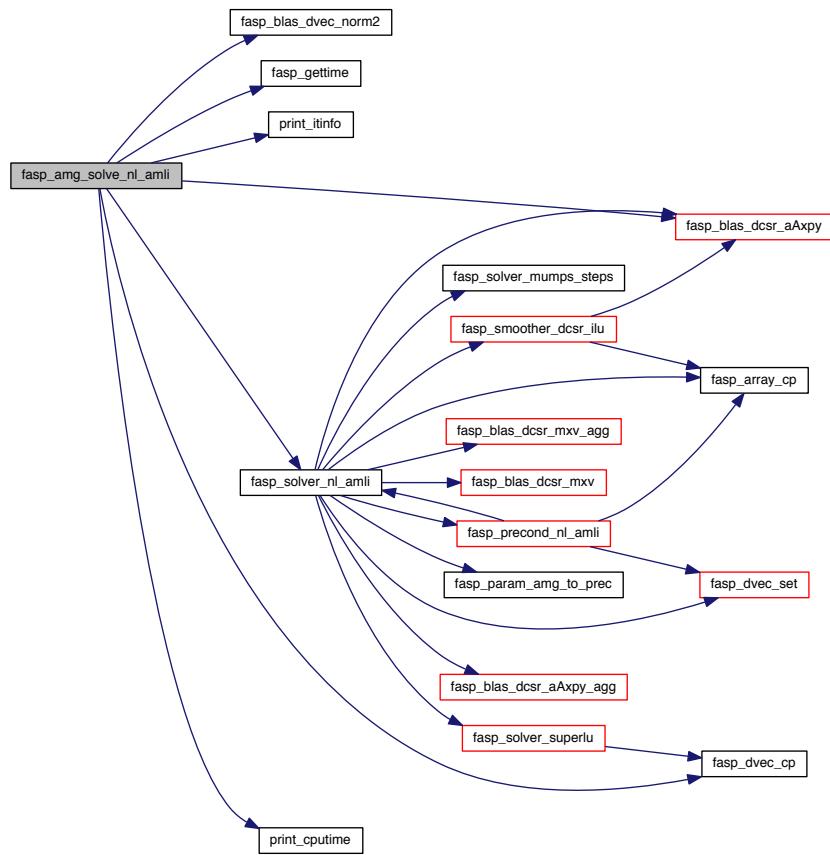
Modified by Chensong 04/21/2013: Fix an output typo

Note

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

Definition at line 209 of file amg_solve.c.

Here is the call graph for this function:



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

FMG – SOLVE phase.

Parameters

<i>mgl</i>	Pointer to AMG data: AMG_data
<i>param</i>	Pointer to AMG parameters: AMG_param

Author

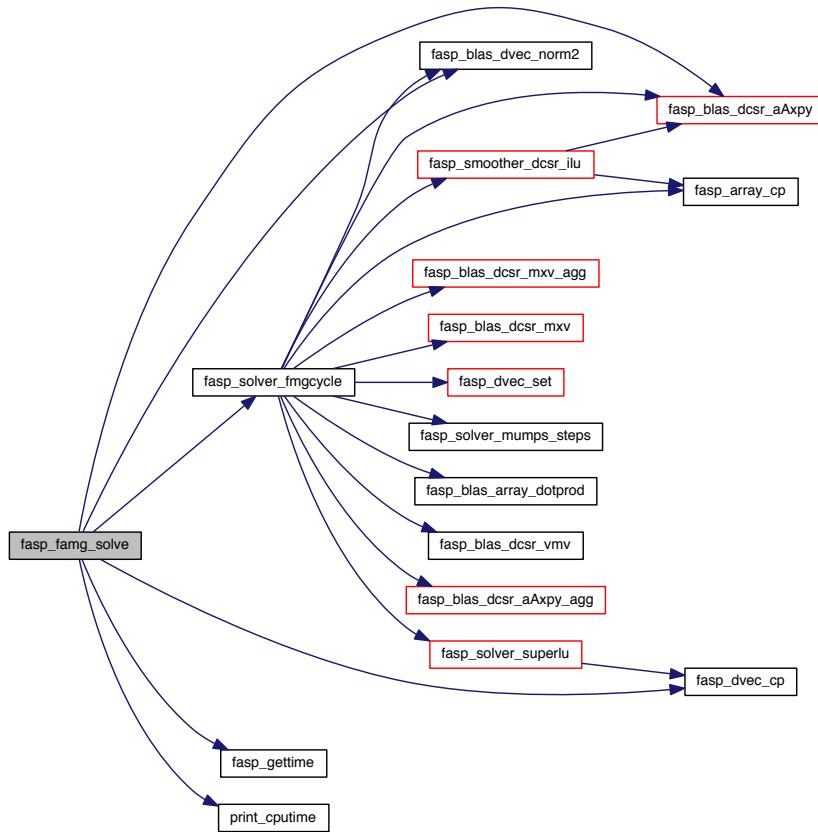
Chensong Zhang

Date

01/10/2012

Definition at line 281 of file amg_solve.c.

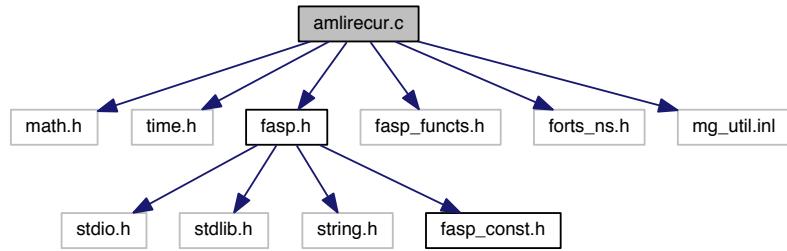
Here is the call graph for this function:



9.7 amlirecur.c File Reference

Abstract AMLI multilevel iteration – recursive version.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "forts_ns.h"
#include "mg_util.inl"
Include dependency graph for amlirecur.c:
```



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.

9.7.1 Detailed Description

Abstract AMLI multilevel iteration – recursive version.

Note

AMLI and non-linear AMLI cycles

9.7.2 Function Documentation

9.7.2.1 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

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

Author

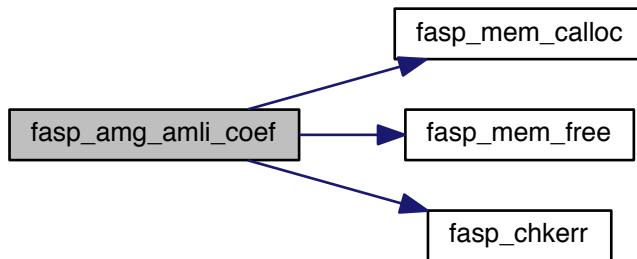
Xiaozhe Hu

Date

01/23/2011

Definition at line 823 of file amlirecur.c.

Here is the call graph for this function:



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

Solve Ax=b with recursive AMLI-cycle.

Parameters

<i>mgl</i>	Pointer to AMG data: AMG_data
<i>param</i>	Pointer to AMG parameters: AMG_param
<i>level</i>	Current level

Author

Xiaozhe Hu

Date

01/23/2011

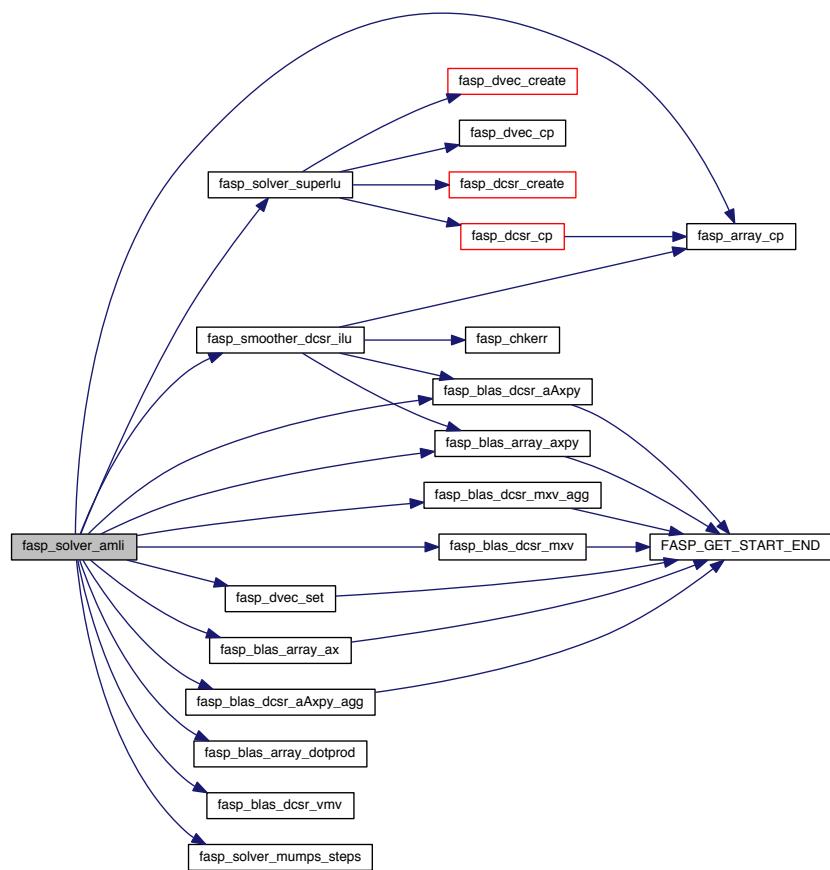
Note

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.

Definition at line 44 of file amlirecur.c.

Here is the call graph for this function:



9.7.2.3 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

<i>mgl</i>	Pointer to AMG_data data
<i>param</i>	Pointer to AMG parameters
<i>level</i>	Current level
<i>num_levels</i>	Total number of levels

Author

Xiaozhe Hu

Date

04/06/2010

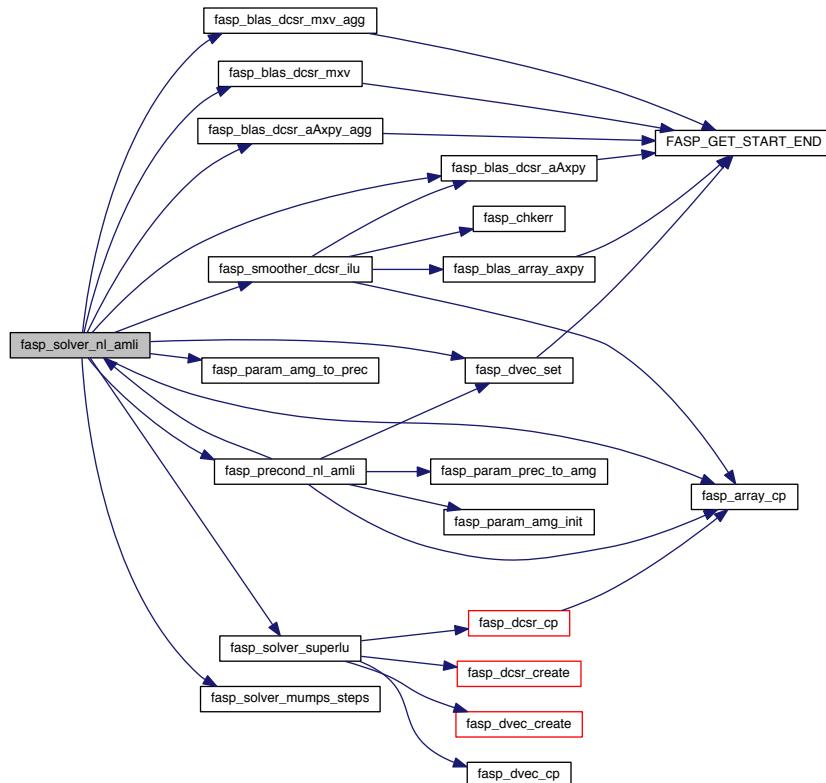
Note

Refer to Xiaozhe Hu, Panayot S. Vassilevski, Jinchao Xu "Comparative Convergence Analysis of Nonlinear AML \leftarrow I-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.

Definition at line 331 of file `amlirecur.c`.

Here is the call graph for this function:



9.7.2.4 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

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

Author

Xiaozhe Hu

Date

04/06/2010

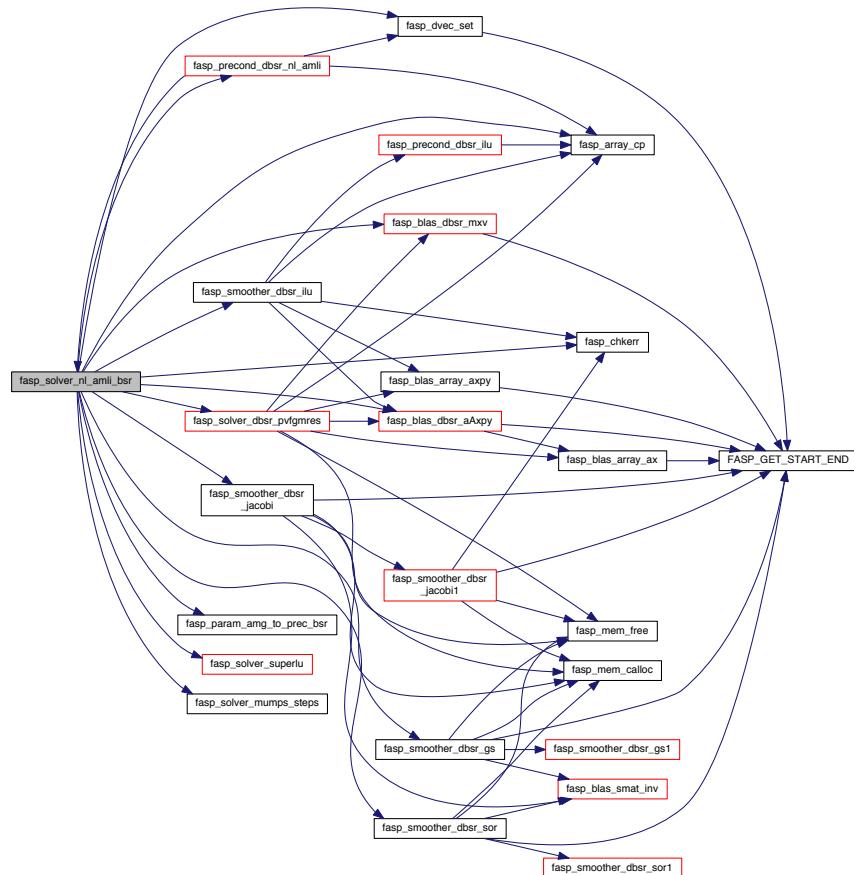
Note

Nonlinear AMLI-cycle. Refer to Xiaozhe 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.

Definition at line 633 of file `amlirecur.c`.

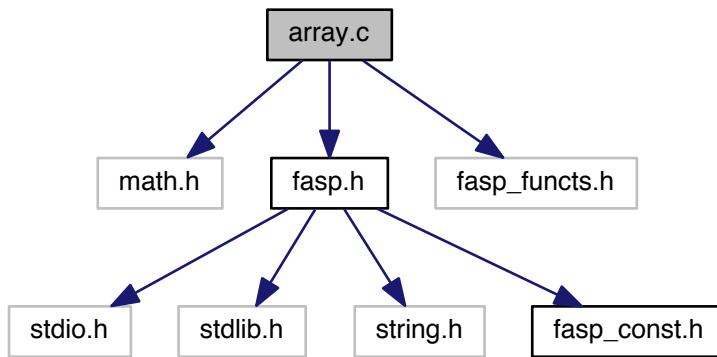
Here is the call graph for this function:



9.8 array.c File Reference

Array operations.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
Include dependency graph for array.c:
```



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.*

9.8.1 Detailed Description

Array operations.

Simple array operations – init, set, copy, etc

9.8.2 Function Documentation

9.8.2.1 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 172 of file array.c.

9.8.2.2 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 212 of file array.c.

9.8.2.3 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 233 of file array.c.

9.8.2.4 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 256 of file array.c.

9.8.2.5 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 32 of file array.c.

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

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

Parameters

<i>n</i>	Number of variables
<i>x</i>	Pointer to the vector
<i>val</i>	Initial value for the REAL array

Author

Chensong Zhang

Date

04/03/2010

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

05/23/2012

Definition at line 52 of file array.c.

Here is the call graph for this function:



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

Copy an array to the other *y*=*x*.

Parameters

<i>n</i>	Number of variables
<i>x</i>	Pointer to the original vector
<i>y</i>	Pointer to the destination vector

Author

Chunsheng Feng, Xiaoqiang Yue

Date

05/23/2012

Definition at line 192 of file array.c.

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

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

Parameters

<i>n</i>	Number of variables
<i>x</i>	Pointer to the vector
<i>val</i>	Initial value for the REAL array

Author

Chensong Zhang

Date

04/03/2010

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

05/25/2012

Definition at line 114 of file array.c.

Here is the call graph for this function:

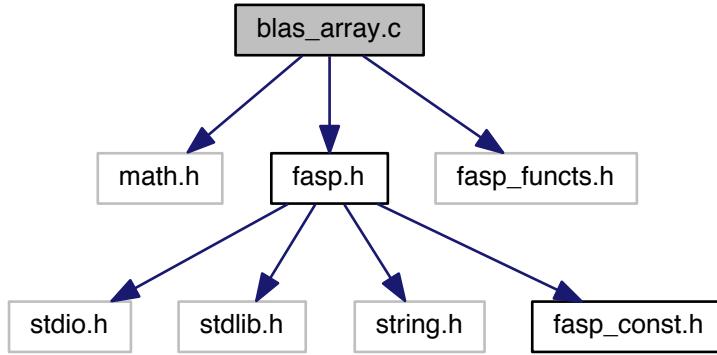


9.9 blas_array.c File Reference

BLAS operations for arrays.

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

Include dependency graph for blas_array.c:



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.

9.9.1 Detailed Description

BLAS operations for arrays.

9.9.2 Function Documentation

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

$x = a*x$

Parameters

<i>n</i>	Number of variables
<i>a</i>	Factor a
<i>x</i>	Pointer to x

Author

Chensong Zhang

Date

07/01/2009

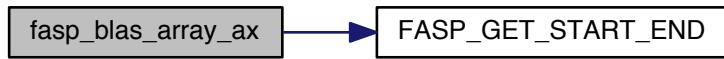
Modified by Chunsheng Feng, Xiaoqiang Yue on 05/23/2012

Note

x is reused to store the resulting array.

Definition at line 35 of file blas_array.c.

Here is the call graph for this function:



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

$$y = a*x + b*y$$

Parameters

<i>n</i>	Number of variables
<i>a</i>	Factor a
<i>x</i>	Pointer to x
<i>b</i>	Factor b
<i>y</i>	Pointer to y

Author

Chensong Zhang

Date

07/01/2009

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.

Here is the call graph for this function:



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

y = *a***x* + *y*

Parameters

<i>n</i>	Number of variables
<i>a</i>	Factor <i>a</i>
<i>x</i>	Pointer to <i>x</i>
<i>y</i>	Pointer to <i>y</i>

Author

Chensong Zhang

Date

07/01/2009

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.

Here is the call graph for this function:



9.9.2.4 void faspblasarray_axpyz(const INT n, const REAL a, REAL * x, REAL * y, REAL * z)

$$z = a*x + y$$

Parameters

<i>n</i>	Number of variables
<i>a</i>	Factor a
<i>x</i>	Pointer to x
<i>y</i>	Pointer to y
<i>z</i>	Pointer to z

Author

Chensong Zhang

Date

07/01/2009

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

Definition at line 167 of file blas_array.c.

Here is the call graph for this function:



9.9.2.5 REAL faspblasarray_dotprod(const INT n, const REAL * x, const REAL * y)

Inner product of two arrays (x,y)

Parameters

<i>n</i>	Number of variables
<i>x</i>	Pointer to x
<i>y</i>	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.

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

L1 norm of array x.

Parameters

<i>n</i>	Number of variables
<i>x</i>	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.

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

L2 norm of array x.

Parameters

<i>n</i>	Number of variables
<i>x</i>	Pointer to <i>x</i>

Returns

L2 norm of *x*

Author

Chensong Zhang

Date

07/01/2009

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

Definition at line 347 of file blas_array.c.

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

Linf norm of array *x*.

Parameters

<i>n</i>	Number of variables
<i>x</i>	Pointer to <i>x</i>

Returns

L_inf norm of *x*

Author

Chensong Zhang

Date

07/01/2009

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

Definition at line 388 of file blas_array.c.

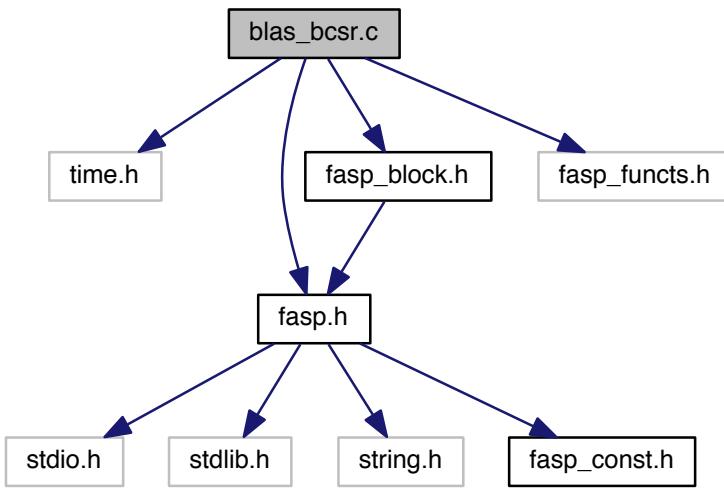
Here is the call graph for this function:



9.10 blas_bcsr.c File Reference

BLAS operations for `block_dCSRmat` matrices.

```
#include <time.h>
#include "fasp.h"
#include "fasp_block.h"
#include "fasp_functs.h"
Include dependency graph for blas_bcsr.c:
```



Functions

- void `fasp_blas_bdcsr_aAxpy` (const `REAL` alpha, `block_dCSRmat` *A, `REAL` *x, `REAL` *y)
Matrix-vector multiplication $y = \text{alpha} \cdot A \cdot x + y$.
- void `fasp_blas_bdcsr_mxv` (`block_dCSRmat` *A, `REAL` *x, `REAL` *y)
Matrix-vector multiplication $y = A \cdot x$.
- void `fasp_blas_bdbsr_aAxpy` (const `REAL` alpha, `block_BSR` *A, `REAL` *x, `REAL` *y)
Matrix-vector multiplication $y = \text{alpha} \cdot A \cdot x + y$.
- void `fasp_blas_bdbsr_mxv` (`block_BSR` *A, `REAL` *x, `REAL` *y)
Matrix-vector multiplication $y = A \cdot x$.

9.10.1 Detailed Description

BLAS operations for `block_dCSRmat` matrices.

9.10.2 Function Documentation

9.10.2.1 void fasp blas_bdbsr_aAxpy (const REAL *alpha*, block_BSR * *A*, REAL * *x*, REAL * *y*)

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

Parameters

<i>alpha</i>	REAL factor a
<i>A</i>	Pointer to block_BSR matrix A
<i>x</i>	Pointer to array x
<i>y</i>	Pointer to array y

Author

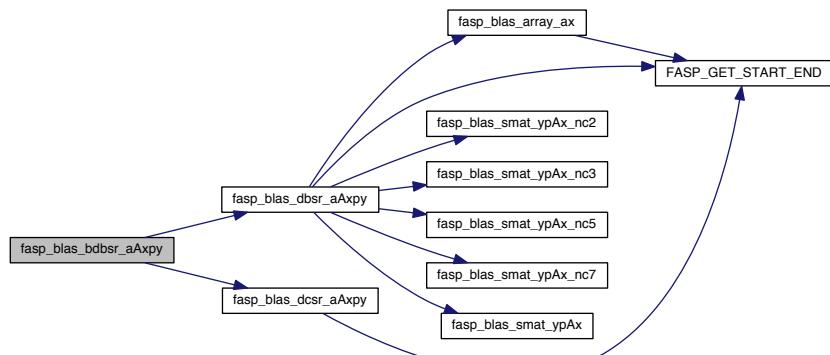
Xiaozhe Hu

Date

11/11/2010

Definition at line 288 of file [blas_bcsr.c](#).

Here is the call graph for this function:



9.10.2.2 void fasp blas bdbsr_mxv ([block_BSR](#) * A, REAL * x, REAL * y)

Matrix-vector multiplication $y = A \cdot x$.

Parameters

<i>A</i>	Pointer to block_BSR matrix A
<i>x</i>	Pointer to array x
<i>y</i>	Pointer to array y

Author

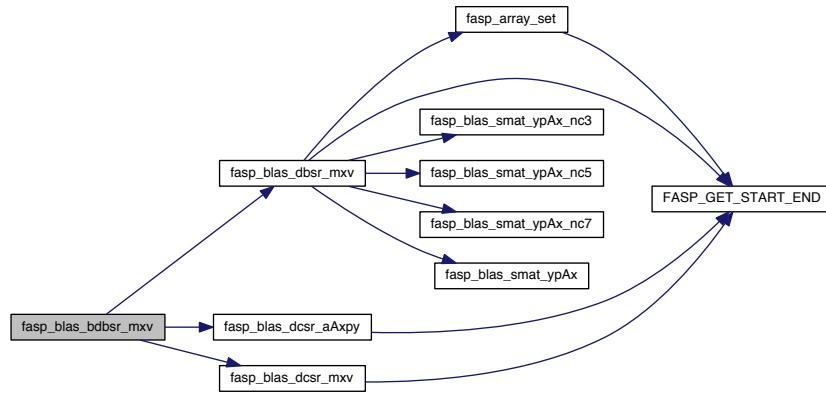
Xiaozhe Hu

Date

11/11/2010

Definition at line 326 of file blas_bcsr.c.

Here is the call graph for this function:

**9.10.2.3 void fasp_blas_bdcsr_aApxy (const REAL alpha, block_dCSRmat * A, REAL * x, REAL * y)**Matrix-vector multiplication $y = \alpha \cdot A \cdot x + y$.**Parameters**

<i>alpha</i>	REAL factor α
<i>A</i>	Pointer to block_dCSRmat matrix A
<i>x</i>	Pointer to array x
<i>y</i>	Pointer to array y

Author

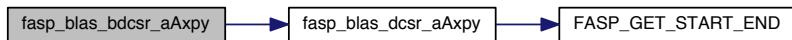
Xiaozhe Hu

Date

06/04/2010

Definition at line 30 of file blas_bcsr.c.

Here is the call graph for this function:



9.10.2.4 void fasp_blas_bdcsr_mxv (**block_dCSRmat** * A, **REAL** * x, **REAL** * y)

Matrix-vector multiplication $y = A \cdot x$.

Parameters

<i>A</i>	Pointer to block_dCSRmat matrix A
<i>x</i>	Pointer to array x
<i>y</i>	Pointer to array y

Author

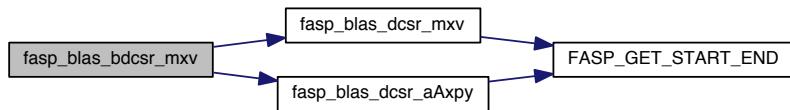
Chensong Zhang

Date

04/27/2013

Definition at line 155 of file **blas_bcsr.c**.

Here is the call graph for this function:

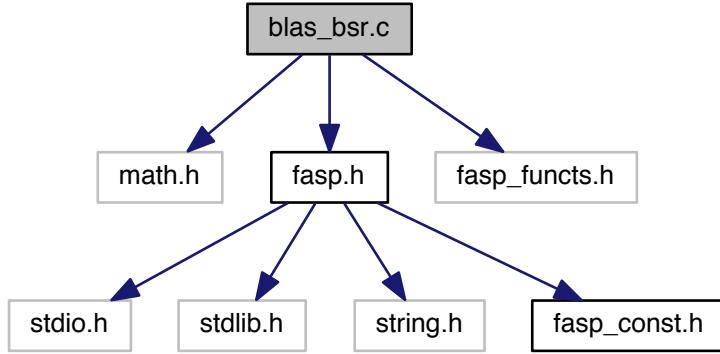


9.11 blas_bsr.c File Reference

BLAS operations for **dBSRmat** matrices.

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

Include dependency graph for blas_bsr.c:



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_aApxy (const REAL alpha, dBSRmat *A, REAL *x, REAL *y)`
*Compute $y := \alpha * A * x + y$.*
- void `fasp_blas_dbsr_aApxy_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!*

9.11.1 Detailed Description

BLAS operations for `dBSRmat` matrices.

9.11.2 Function Documentation

9.11.2.1 void fasp_blas_dbsr_aAxpby (const REAL *alpha*, dBsrmat * *A*, REAL * *x*, const REAL *beta*, REAL * *y*)

Compute $y := \alpha \cdot A \cdot x + \beta \cdot y$.

Parameters

<i>alpha</i>	REAL factor alpha
<i>A</i>	Pointer to the <code>dBSRmat</code> matrix
<i>x</i>	Pointer to the array <i>x</i>
<i>beta</i>	REAL factor beta
<i>y</i>	Pointer to the array <i>y</i>

Author

Zhiyang Zhou

Date

10/25/2010

Modified by Chunsheng Feng, Zheng Li

Date

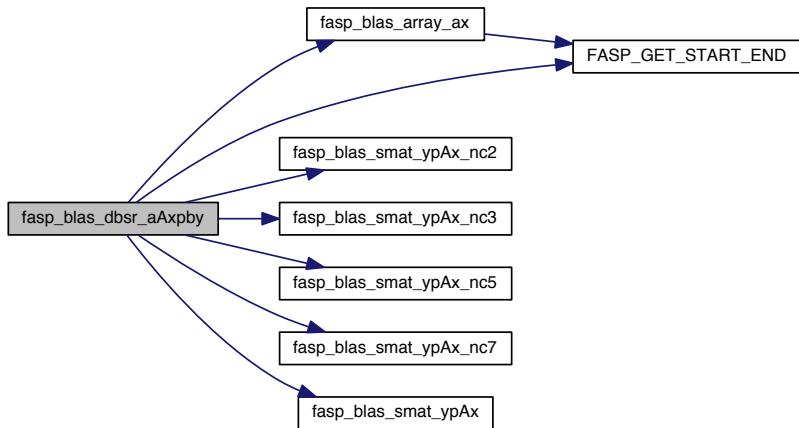
06/29/2012

Note

Works for general nb (Xiaozhe)

Definition at line 60 of file `blas_bsr.c`.

Here is the call graph for this function:



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

Compute $y := \alpha \cdot A \cdot x + y$.

Parameters

<i>alpha</i>	REAL factor alpha
<i>A</i>	Pointer to the <code>dBSRmat</code> matrix
<i>x</i>	Pointer to the array <i>x</i>
<i>y</i>	Pointer to the array <i>y</i>

Author

Zhiyang Zhou

Date

10/25/2010

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

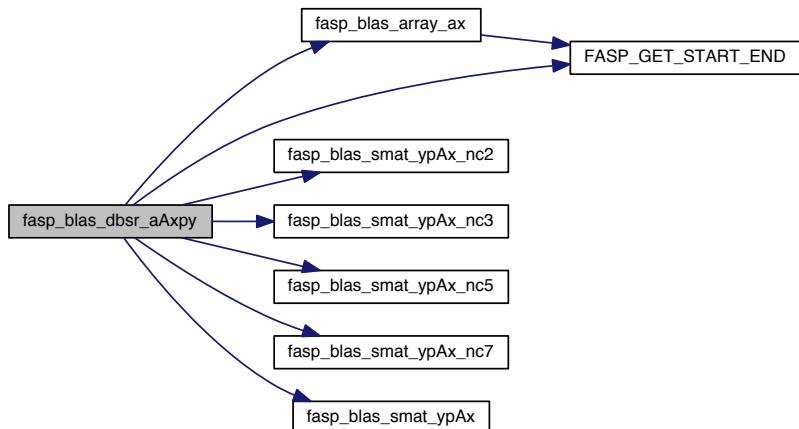
05/23/2012

Note

Works for general nb (Xiaozhe)

Definition at line 339 of file `blas_bsr.c`.

Here is the call graph for this function:



9.11.2.3 void fasp blas dbsr_aApxy_agg (const REAL *alpha*, `dBSRmat` * *A*, REAL * *x*, REAL * *y*)

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

Parameters

<i>alpha</i>	REAL factor alpha
<i>A</i>	Pointer to the <code>dBSRmat</code> matrix
<i>x</i>	Pointer to the array <i>x</i>
<i>y</i>	Pointer to the array <i>y</i>

Author

Xiaozhe Hu

Date

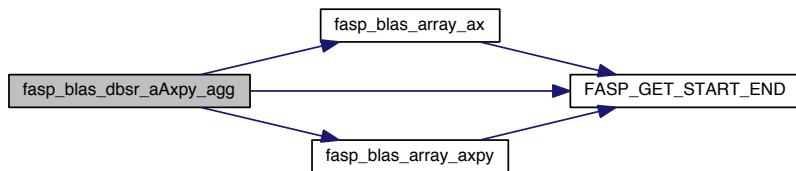
01/02/2014

Note

Works for general nb (Xiaozhe)

Definition at line 612 of file `blas_bsr.c`.

Here is the call graph for this function:



9.11.2.4 void `fasp_bla_dbsr_axm` (`dBSRmat * A`, `const REAL alpha`)

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

Parameters

<i>A</i>	Pointer to <code>dBSRmat</code> matrix <i>A</i>
<i>alpha</i>	REAL factor <i>alpha</i>

Author

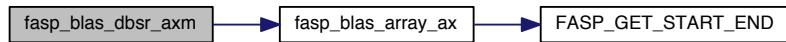
Xiaozhe Hu

Date

05/26/2014

Definition at line 30 of file blas_bsr.c.

Here is the call graph for this function:

**9.11.2.5 void fasp_bla...mxm (dB...mat * A, dB...mat * B, dB...mat * C)**

Sparse matrix multiplication C=A*B.

Parameters

<i>A</i>	Pointer to the <code>dB...mat</code> matrix A
<i>B</i>	Pointer to the <code>dB...mat</code> matrix B
<i>C</i>	Pointer to <code>dB...mat</code> matrix equal to A*B

Author

Xiaozhe Hu

Date

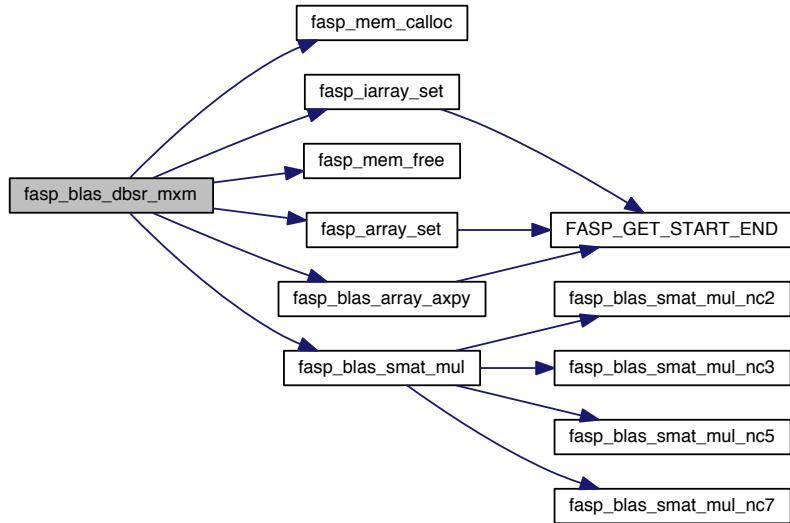
05/26/2014

Note

This fct will be replaced! – Xiaozhe

Definition at line 4594 of file blas_bsr.c.

Here is the call graph for this function:



9.11.2.6 void fasp_bla..._mxv (dB...mat * A, REAL * x, REAL * y)

Compute $y := A*x$.

Parameters

A	Pointer to the <code>dB...mat</code> matrix
x	Pointer to the array x
y	Pointer to the array y

Author

Zhiyang Zhou

Date

10/25/2010

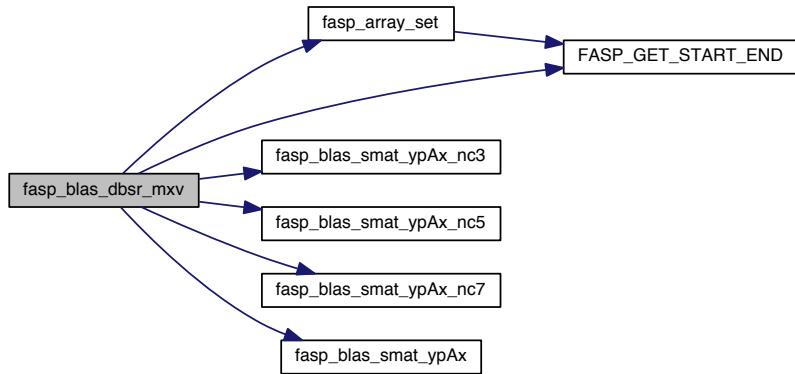
Note

Works for general nb (Xiaozhe)

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

Definition at line 897 of file blas_bsr.c.

Here is the call graph for this function:



9.11.2.7 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

<i>A</i>	Pointer to the dBSRmat matrix
<i>x</i>	Pointer to the array <i>x</i>
<i>y</i>	Pointer to the array <i>y</i>

Author

Xiaozhe Hu

Date

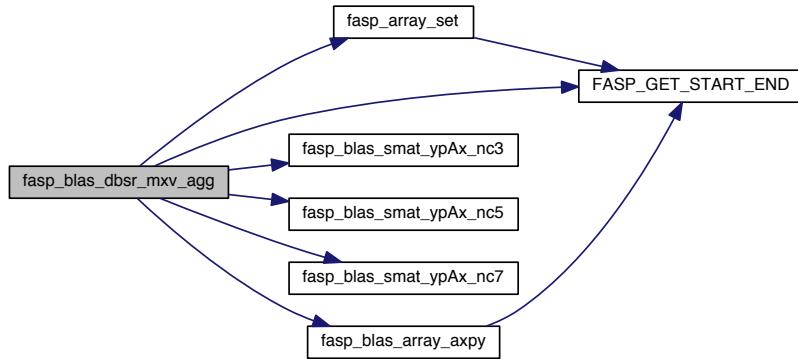
01/02/2014

Note

Works for general nb (Xiaozhe)

Definition at line 2643 of file blas_bsr.c.

Here is the call graph for this function:



9.11.2.8 void fasp_blas_dbsr_rap (dBsrmat * R, dBsrmat * A, dBsrmat * P, dBsrmat * B)

dBSRmat sparse matrix multiplication $B=R \cdot A \cdot P$

Parameters

<i>R</i>	Pointer to the dBSRmat matrix
<i>A</i>	Pointer to the dBSRmat matrix
<i>P</i>	Pointer to the dBSRmat matrix
<i>B</i>	Pointer to dBSRmat matrix equal to $R \cdot A \cdot P$ (output)

Author

Xiaozhe Hu, Chunsheng Feng, Zheng Li

Date

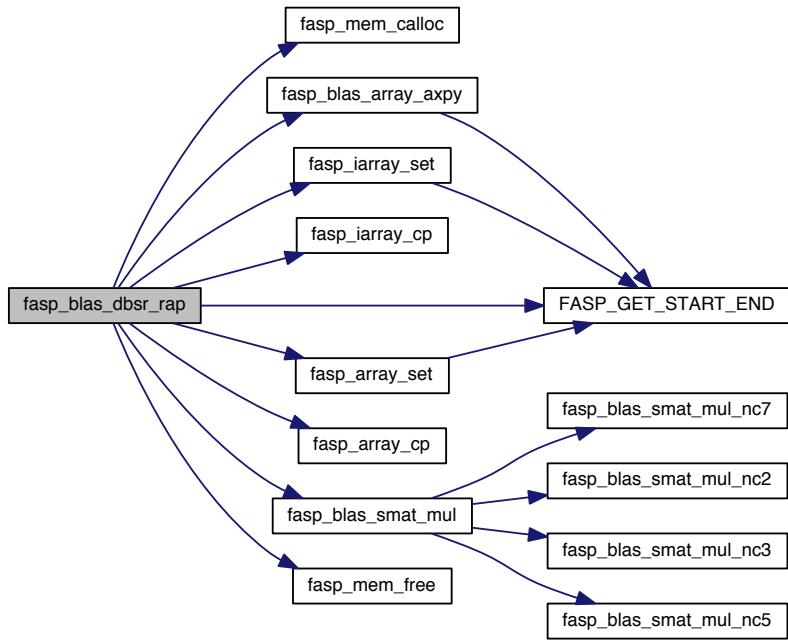
10/24/2012

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 4898 of file blas_bsr.c.

Here is the call graph for this function:



9.11.2.9 void fasp_blas_dbsr_rap1 (dBsrmat * R, dBsrmat * A, dBsrmat * P, dBsrmat * B)

dBSRmat sparse matrix multiplication $B=R \cdot A \cdot P$

Parameters

<i>R</i>	Pointer to the dBSRmat matrix
<i>A</i>	Pointer to the dBSRmat matrix
<i>P</i>	Pointer to the dBSRmat matrix
<i>B</i>	Pointer to dBSRmat matrix equal to $R \cdot A \cdot P$ (output)

Author

Chunsheng Feng, Xiaoqiang Yue and Xiaozhe Hu

Date

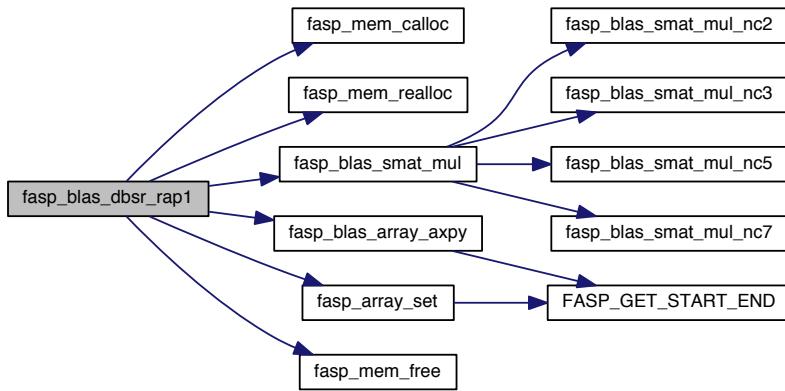
08/08/2011

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 4714 of file blas_bsr.c.

Here is the call graph for this function:



9.11.2.10 void fasp_bla_dbsr_rap_agg (dBsrmat * R, dBsrmat * A, dBsrmat * P, dBsrmat * B)

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

Parameters

<i>R</i>	Pointer to the dBSRmat matrix
<i>A</i>	Pointer to the dBSRmat matrix
<i>P</i>	Pointer to the dBSRmat matrix
<i>B</i>	Pointer to dBSRmat matrix equal to $R \cdot A \cdot P$ (output)

Author

Xiaozhe Hu

Date

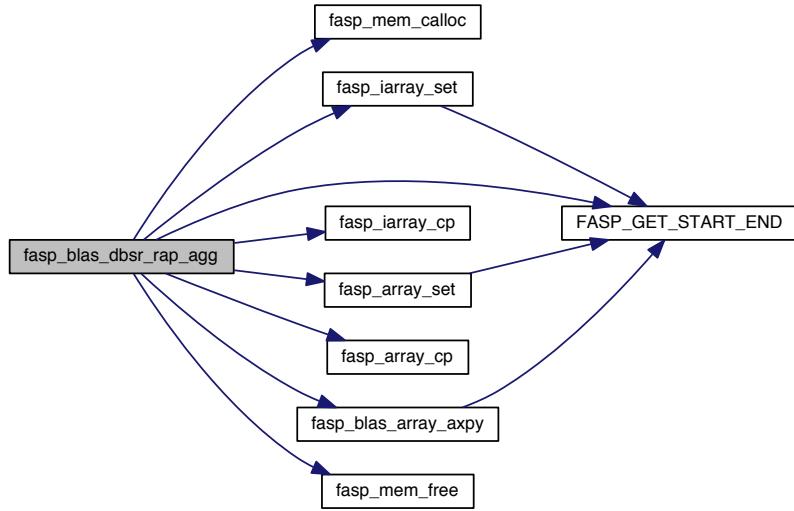
10/24/2012

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 5163 of file blas_bsr.c.

Here is the call graph for this function:

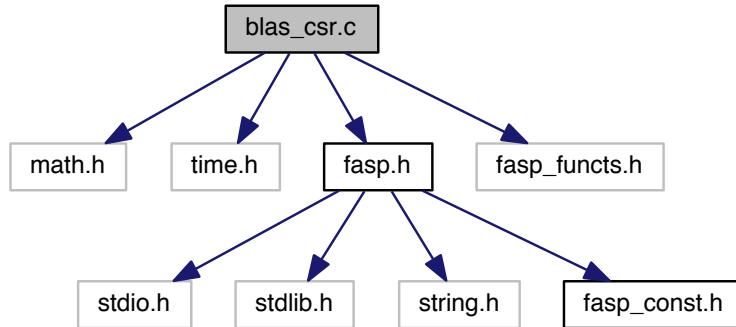


9.12 blas_csr.c File Reference

BLAS operations for [dCSRmat](#) matrices.

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

Include dependency graph for blas_csr.c:



Functions

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

$$\text{compute } C = \alpha * A + \beta * B \text{ in CSR format}$$
- `void fasp_blas_dcsr_axm (dCSRmat *A, const REAL alpha)`

$$\text{Multiply a sparse matrix } A \text{ in CSR format by a scalar } \alpha.$$
- `void fasp_blas_dcsr_mxv (dCSRmat *A, REAL *x, REAL *y)`

$$\text{Matrix-vector multiplication } y = A * x.$$
- `void fasp_blas_dcsr_mxv_agg (dCSRmat *A, REAL *x, REAL *y)`

$$\text{Matrix-vector multiplication } y = A * x, \text{ where the entries of } A \text{ are all ones.}$$
- `void fasp_blas_dcsr_aAxpy (const REAL alpha, dCSRmat *A, REAL *x, REAL *y)`

$$\text{Matrix-vector multiplication } y = \alpha * A * x + y.$$
- `void fasp_blas_dcsr_aAxpy_agg (const REAL alpha, dCSRmat *A, REAL *x, REAL *y)`

$$\text{Matrix-vector multiplication } y = \alpha * A * x + y \text{ (the entries of } A \text{ are all ones)}$$
- `REAL fasp_blas_dcsr_vmv (dCSRmat *A, REAL *x, REAL *y)`

$$\text{vector-Matrix-vector multiplication } \alpha = y' * A * x$$
- `void fasp_blas_dcsr_mxm (dCSRmat *A, dCSRmat *B, dCSRmat *C)`

$$\text{Sparse matrix multiplication } C = A * B.$$
- `void fasp_blas_dcsr_rap (dCSRmat *R, dCSRmat *A, dCSRmat *P, dCSRmat *RAP)`

$$\text{Triple sparse matrix multiplication } B = R * A * P.$$
- `void fasp_blas_dcsr_rap_agg (dCSRmat *R, dCSRmat *A, dCSRmat *P, dCSRmat *RAP)`

$$\text{Triple sparse matrix multiplication } B = R * A * P.$$
- `void fasp_blas_dcsr_rap_agg1 (dCSRmat *R, dCSRmat *A, dCSRmat *P, dCSRmat *B)`

$$\text{Triple sparse matrix multiplication } B = R * A * P \text{ (nonzero entries of } R \text{ and } P \text{ are ones)}$$
- `void fasp_blas_dcsr_ptap (dCSRmat *Pt, dCSRmat *A, dCSRmat *P, dCSRmat *Ac)`

$$\text{Triple sparse matrix multiplication } B = P' * A * P.$$
- `void fasp_blas_dcsr_rap4 (dCSRmat *R, dCSRmat *A, dCSRmat *P, dCSRmat *B, INT *icor_ysk)`

$$\text{Triple sparse matrix multiplication } B = R * A * P.$$

9.12.1 Detailed Description

BLAS operations for `dCSRmat` matrices.

Note

Sparse functions usually contain three runs. The three runs are all the same but they 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.

9.12.2 Function Documentation

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

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

Parameters

<i>alpha</i>	REAL factor alpha
<i>A</i>	Pointer to dCSRmat matrix A
<i>x</i>	Pointer to array x
<i>y</i>	Pointer to array y

Author

Chensong Zhang

Date

07/01/2009

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

Definition at line 480 of file `blas_csr.c`.

Here is the call graph for this function:



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

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

Parameters

<i>alpha</i>	REAL factor alpha
<i>A</i>	Pointer to dCSRmat matrix A
<i>x</i>	Pointer to array x
<i>y</i>	Pointer to array y

Author

Xiaozhe Hu

Date

02/22/2011

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

Definition at line 594 of file `blas_csr.c`.

Here is the call graph for this function:



9.12.2.3 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

<i>A</i>	Pointer to <code>dCSRmat</code> matrix
<i>alpha</i>	REAL factor alpha
<i>B</i>	Pointer to <code>dCSRmat</code> matrix
<i>beta</i>	REAL factor beta
<i>C</i>	Pointer to <code>dCSRmat</code> matrix

Returns

`FASP_SUCCESS` if succeed, `ERROR` if not

Author

Xiaozhe Hu

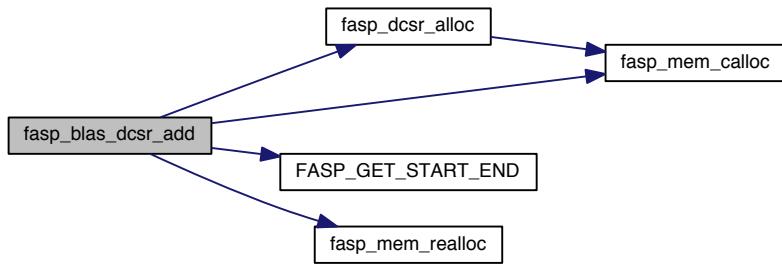
Date

11/07/2009

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

Definition at line 48 of file `blas_csr.c`.

Here is the call graph for this function:



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

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

Parameters

<i>A</i>	Pointer to dCSRmat matrix <i>A</i>
<i>alpha</i>	REAL factor <i>alpha</i>

Author

Chensong Zhang

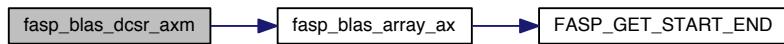
Date

07/01/2009

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

Definition at line 201 of file `blas_csr.c`.

Here is the call graph for this function:



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

Sparse matrix multiplication *C*=*A***B*.

Parameters

<i>A</i>	Pointer to the dCSRmat matrix <i>A</i>
<i>B</i>	Pointer to the dCSRmat matrix <i>B</i>
<i>C</i>	Pointer to dCSRmat matrix equal to <i>A</i> * <i>B</i>

Author

Xiaozhe Hu

Date

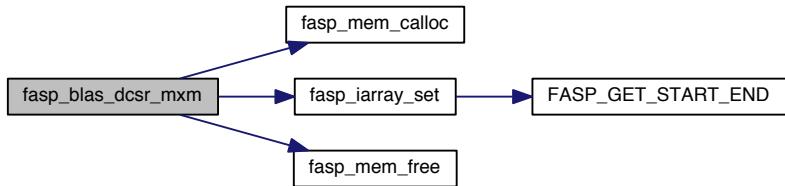
11/07/2009

Note

This fct will be replaced! –Chensong

Definition at line 760 of file blas_csr.c.

Here is the call graph for this function:



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

Matrix-vector multiplication $y = A \cdot x$.

Parameters

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

Author

Chensong Zhang

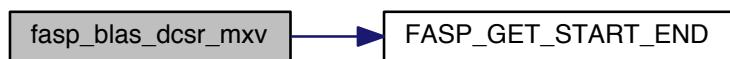
Date

07/01/2009

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

Definition at line 225 of file blas_csr.c.

Here is the call graph for this function:



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

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

Parameters

<i>A</i>	Pointer to <code>dCSRmat</code> matrix A
<i>x</i>	Pointer to array x
<i>y</i>	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`.

Here is the call graph for this function:

**9.12.2.8 void fasp_blas_dcsr_ptap (`dCSRmat` * *Pt*, `dCSRmat` * *A*, `dCSRmat` * *P*, `dCSRmat` * *Ac*)**Triple sparse matrix multiplication $B=P^*A*P$.**Parameters**

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

Author

Ludmil Zikatanov, Chensong Zhang

Date

05/10/2010

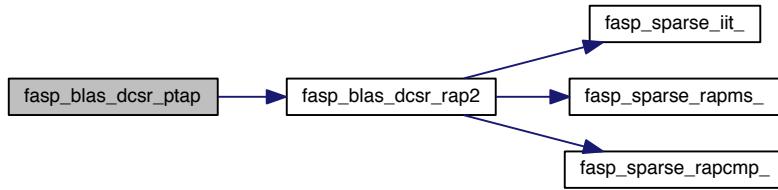
Modified by Chunsheng Feng, Zheng Li on 10/19/2012

Note

Driver to compute triple matrix product $P^T * A * P$ using Itz CSR format. In Itz 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_itz[k] = ia_usual[k]+1, ja_itz[k] = ja_usual[k]+1, a_itz[k] = a_usual[k].

Definition at line 1598 of file blas_csr.c.

Here is the call graph for this function:



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

Triple sparse matrix multiplication $B=R*A*P$.

Parameters

<i>R</i>	Pointer to the dCSRmat matrix R
<i>A</i>	Pointer to the dCSRmat matrix A
<i>P</i>	Pointer to the dCSRmat matrix P
<i>RAP</i>	Pointer to dCSRmat matrix equal to $R*A*P$

Author

Xuehai Huang, Chensong Zhang

Date

05/10/2010

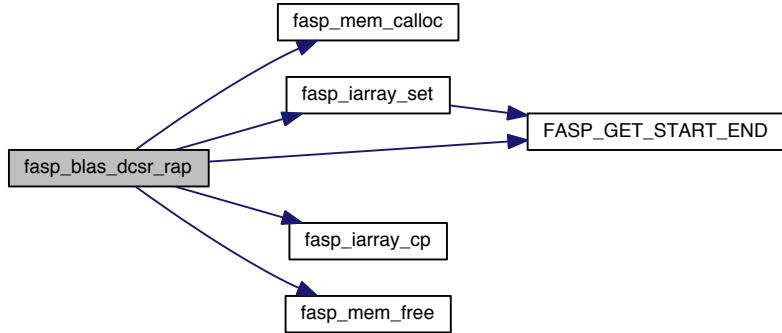
Modified by Chunsheng Feng, Xiaoqiang Yue on 05/26/2012

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 867 of file blas_csr.c.

Here is the call graph for this function:



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

Triple sparse matrix multiplication $B=R \cdot A \cdot P$.

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 $R \cdot A \cdot P$
$icor_ysk$	pointer to the array

Author

Feng Chunsheng, Yue Xiaoqiang

Date

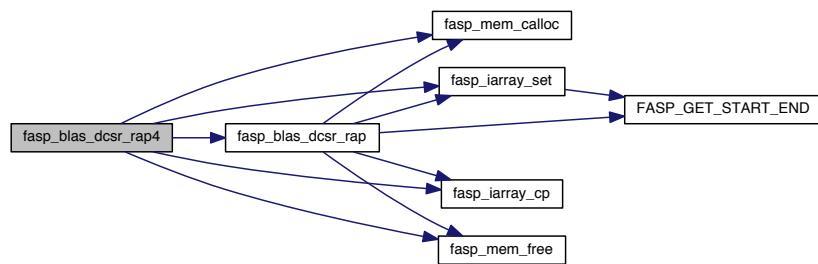
08/02/2011

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 1700 of file blas_csr.c.

Here is the call graph for this function:



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

Triple sparse matrix multiplication $B=R \cdot A \cdot P$.

Parameters

<i>R</i>	Pointer to the dCSRmat matrix <i>R</i>
<i>A</i>	Pointer to the dCSRmat matrix <i>A</i>
<i>P</i>	Pointer to the dCSRmat matrix <i>P</i>
<i>RAP</i>	Pointer to dCSRmat matrix equal to $R \cdot A \cdot P$

Author

Xiaozhe Hu

Date

05/10/2010

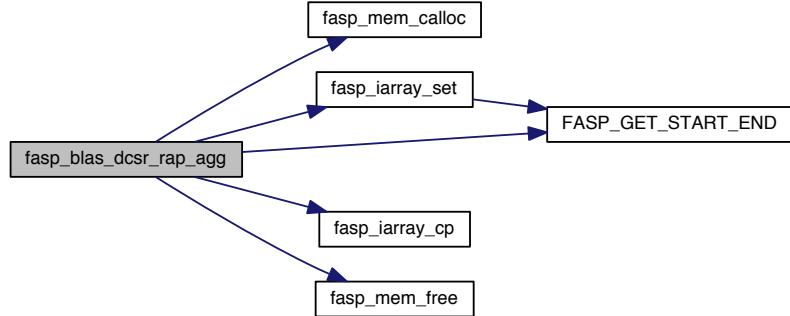
Modified by Chunsheng Feng, Xiaoqiang Yue on 05/26/2012

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 1150 of file blas_csr.c.

Here is the call graph for this function:



9.12.2.12 void fasp_blas_dcsr_rap_agg(dCSRmat * R, dCSRmat * A, dCSRmat * P, dCSRmat * B)

Triple sparse matrix multiplication $B=R \cdot A \cdot P$ (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 $R \cdot A \cdot P$

Author

Xiaozhe Hu

Date

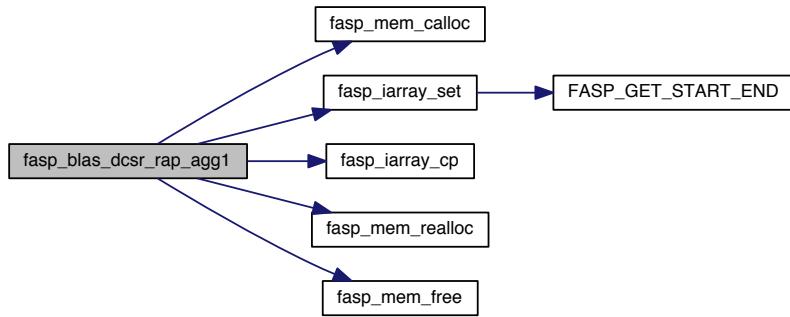
02/21/2011

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 1415 of file blas_csr.c.

Here is the call graph for this function:



9.12.2.13 REAL fasp_bias_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

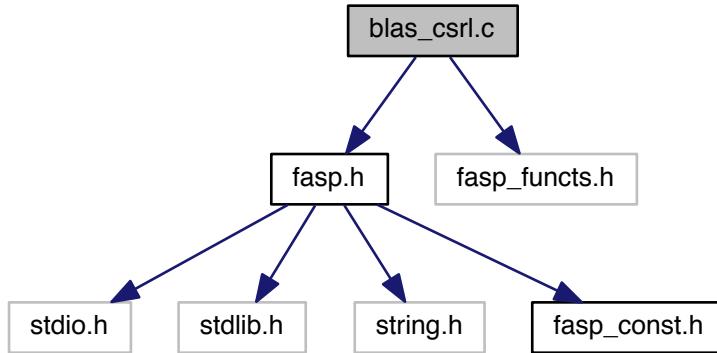
Definition at line 705 of file blas_csr.c.

9.13 blas_csr1.c File Reference

BLAS operations for [dCSRmat](#) matrices.

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

Include dependency graph for blas_csrl.c:



Functions

- void `fasp_blas_dcsrl_mxv (dCSRmat *A, REAL *x, REAL *y)`
*Compute $y = A*x$ for a sparse matrix in CSRL format.*

9.13.1 Detailed Description

BLAS operations for `dCSRmat` 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.

9.13.2 Function Documentation

9.13.2.1 void `fasp_blas_dcsrl_mxv (dCSRmat * A, REAL * x, REAL * y)`

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

Parameters

<code>A</code>	Pointer to <code>dCSRmat</code> matrix <code>A</code>
<code>x</code>	Pointer to <code>REAL</code> array of vector <code>x</code>
<code>y</code>	Pointer to <code>REAL</code> array of vector <code>y</code>

Date

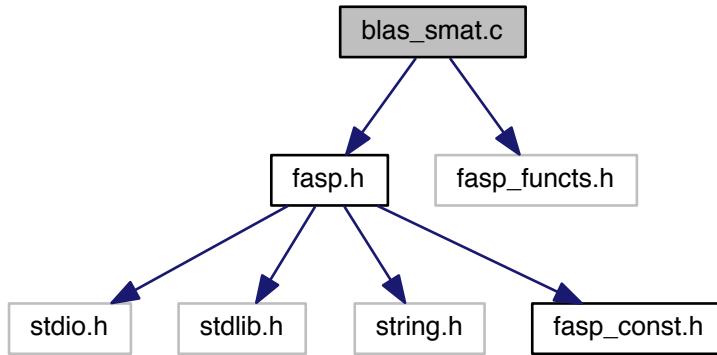
2011/01/07

Definition at line 28 of file `blas_csrl.c`.

9.14 blas_smat.c File Reference

BLAS operations for *small* dense matrix.

```
#include "fasp.h"
#include "fasp_functs.h"
Include dependency graph for blas_smat.c:
```



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)`

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

*Compute the matrix product of two 7*7 matrices a and b, stored in c.*
- void `fasp_blas_array_axpyz_nc2 (REAL a, REAL *x, REAL *y, REAL *z)`

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

$$y = a*x + y, \text{ the length of } x \text{ and } y \text{ is } 2$$
- void `fasp_blas_array_axpy_nc3 (const REAL a, REAL *x, REAL *y)`

$$y = a*x + y, \text{ the length of } x \text{ and } y \text{ is } 3$$
- void `fasp_blas_array_axpy_nc5 (const REAL a, REAL *x, REAL *y)`

$$y = a*x + y, \text{ the length of } x \text{ and } y \text{ is } 5$$
- void `fasp_blas_array_axpy_nc7 (const REAL a, REAL *x, REAL *y)`

$$y = a*x + y, \text{ the length of } x \text{ and } y \text{ is } 7$$
- 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, INT n)`

*Compute $y := y - Ax$, where 'A' is a $n*n$ dense matrix.*
- void `fasp_blas_smat_aAxpb (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)`
 $y := y - \text{Ass} \cdot x$, where ' A' is a 7×7 dense matrix, Ass is its saturation part 6×6 .
- void `fasp_blas_smat_ymAx_ns (REAL *A, REAL *x, REAL *y, const INT n)`
 $y := y - \text{Ass} \cdot x$, where ' A' is a $n \times n$ dense matrix, Ass is its saturation part $(n-1) \times (n-1)$.

9.14.1 Detailed Description

BLAS operations for *small* dense matrix.

9.14.2 Function Documentation

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

$y = a \cdot x + y$, 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 683 of file `blas_smat.c`.

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

$y = a \cdot x + y$, 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 706 of file `blas_smat.c`.

9.14.2.3 void fasp_blas_array_axpy_nc5 (const REAL a, REAL * x, REAL * y)

$y = a \cdot x + y$, the length of x and y is 5

Parameters

<i>a</i>	REAL factor a
<i>x</i>	Pointer to the original array
<i>y</i>	Pointer to the destination array

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 735 of file blas_smat.c.

9.14.2.4 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

<i>a</i>	REAL factor a
<i>x</i>	Pointer to the original array
<i>y</i>	Pointer to the destination array

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 782 of file blas_smat.c.

9.14.2.5 void fasp_blas_array_axpyz_nc2 (REAL a, REAL * x, REAL * y, REAL * z)

$z = a*x + y$

Parameters

<i>a</i>	REAL factor a
<i>x</i>	Pointer to the original array 1
<i>y</i>	Pointer to the original array 2
<i>z</i>	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 498 of file blas_smat.c.

9.14.2.6 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 525 of file blas_smat.c.

9.14.2.7 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 558 of file blas_smat.c.

9.14.2.8 void fasp blas_array_axpyz_nc7(const REAL a, REAL * x, REAL * y, REAL * z)

$Z = a*x + y$

Parameters

<i>a</i>	REAL factor a
<i>x</i>	Pointer to the original array 1
<i>y</i>	Pointer to the original array 2
<i>z</i>	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 609 of file blas_smat.c.

9.14.2.9 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

<i>alpha</i>	REAL factor alpha
<i>A</i>	Pointer to the REAL array which stands for a n*n full matrix
<i>x</i>	Pointer to the REAL array with length n
<i>beta</i>	REAL factor beta
<i>y</i>	Pointer to the REAL array with length n
<i>n</i>	Length of array <i>x</i> and <i>y</i>

Author

Zhiyang Zhou

Date

2010/10/25

Definition at line 1306 of file blas_smat.c.

9.14.2.10 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

<i>a</i>	Pointer to the REAL array which stands a n*n matrix
<i>b</i>	Pointer to the REAL array which stands a n*n matrix
<i>n</i>	Dimension of the matrix
<i>alpha</i>	Scalar
<i>beta</i>	Scalar
<i>c</i>	Pointer to the REAL array which stands a n*n matrix

Author

Xiaozhe Hu

Date

05/26/2014

Definition at line 52 of file blas_smat.c.

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

Compute alpha*a, store in a.

Parameters

<i>a</i>	Pointer to the REAL array which stands a n*n matrix
<i>n</i>	Dimension of the matrix
<i>alpha</i>	Scalar

Author

Xiaozhe Hu

Date

05/26/2014

Definition at line 24 of file blas_smat.c.

9.14.2.12 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

<i>a</i>	Pointer to the REAL array which stands a n*n matrix
<i>b</i>	Pointer to the REAL array which stands a n*n matrix
<i>c</i>	Pointer to the REAL array which stands a n*n matrix
<i>n</i>	Dimension of the matrix

Author

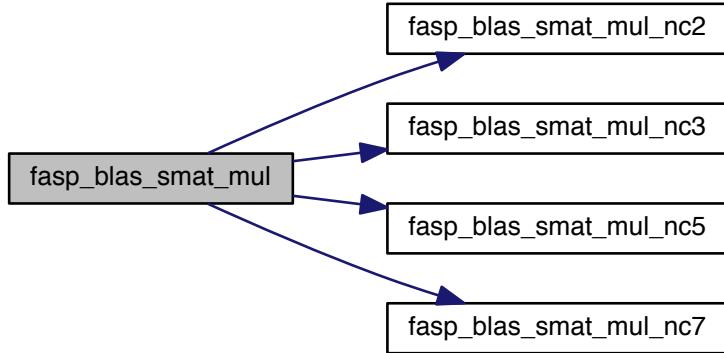
Xiaozhe Hu, Shiquan Zhang

Date

04/21/2010

Definition at line 446 of file blas_smat.c.

Here is the call graph for this function:



9.14.2.13 void faspblas_smat_mul_nc2 (REAL * a, REAL * b, REAL * c)

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

Parameters

<i>a</i>	Pointer to the REAL array which stands a n*n matrix
<i>b</i>	Pointer to the REAL array which stands a n*n matrix
<i>c</i>	Pointer to the REAL array which stands a n*n matrix

Author

Xiaozhe Hu

Date

18/11/2011

Definition at line 231 of file blas_smat.c.

9.14.2.14 void faspblas_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

<i>a</i>	Pointer to the REAL array which stands a n*n matrix
<i>b</i>	Pointer to the REAL array which stands a n*n matrix
<i>c</i>	Pointer to the REAL array which stands a n*n matrix

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 260 of file blas_smat.c.

9.14.2.15 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

<i>a</i>	Pointer to the REAL array which stands a 5*5 matrix
<i>b</i>	Pointer to the REAL array which stands a 5*5 matrix
<i>c</i>	Pointer to the REAL array which stands a 5*5 matrix

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 297 of file blas_smat.c.

9.14.2.16 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

<i>a</i>	Pointer to the REAL array which stands a 7*7 matrix
<i>b</i>	Pointer to the REAL array which stands a 7*7 matrix
<i>c</i>	Pointer to the REAL array which stands a 7*7 matrix

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 356 of file blas_smat.c.

9.14.2.17 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

<i>a</i>	Pointer to the REAL array which stands a n*n matrix
<i>b</i>	Pointer to the REAL array with length n
<i>c</i>	Pointer to the REAL array with length n
<i>n</i>	Dimension of the matrix

Author

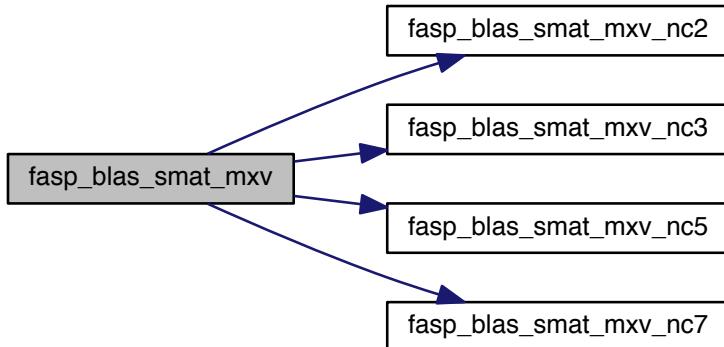
Xiaozhe Hu, Shiquan Zhang

Date

04/21/2010

Definition at line 181 of file blas_smat.c.

Here is the call graph for this function:



9.14.2.18 void faspblas_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

<i>a</i>	Pointer to the REAL array which stands a 2*2 matrix
<i>b</i>	Pointer to the REAL array with length 2
<i>c</i>	Pointer to the REAL array with length 2

Author

Xiaozhe Hu

Date

18/11/2010

Definition at line 81 of file blas_smat.c.

9.14.2.19 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

<i>a</i>	Pointer to the REAL array which stands a 3*3 matrix
<i>b</i>	Pointer to the REAL array with length 3
<i>c</i>	Pointer to the REAL array with length 3

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 103 of file blas_smat.c.

9.14.2.20 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

<i>a</i>	Pointer to the REAL array which stands a 5*5 matrix
<i>b</i>	Pointer to the REAL array with length 5
<i>c</i>	Pointer to the REAL array with length 5

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 126 of file blas_smat.c.

9.14.2.21 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

<i>a</i>	Pointer to the REAL array which stands a 7*7 matrix
<i>b</i>	Pointer to the REAL array with length 7
<i>c</i>	Pointer to the REAL array with length 7

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 152 of file blas_smat.c.

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

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

Parameters

<i>A</i>	Pointer to the $n \times n$ dense matrix
<i>x</i>	Pointer to the REAL array with length n
<i>y</i>	Pointer to the REAL array with length n
<i>n</i>	the dimension of the dense matrix

Author

Zhiyang Zhou, Xiaozhe Hu

Date

2010/10/25

Definition at line 1205 of file blas_smat.c.

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

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

Parameters

<i>A</i>	Pointer to the 2×2 dense matrix
<i>x</i>	Pointer to the REAL array with length 3
<i>y</i>	Pointer to the REAL array with length 3

Author

Xiaozhe Hu

Date

18/11/2011

Note

Works for 2-component

Definition at line 1075 of file blas_smat.c.

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

Compute $y := y - Ax$, where 'A' is a $n \times n$ 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 1103 of file blas_smat.c.

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

Compute $y := y - Ax$, where 'A' is a $n \times n$ 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 1133 of file blas_smat.c.

9.14.2.26 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

<i>A</i>	Pointer to the 7*7 dense matrix
<i>x</i>	Pointer to the REAL array with length 7
<i>y</i>	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 1167 of file blas_smat.c.

9.14.2.27 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 saturation part $(n-1)*(n-1)$.

Parameters

<i>A</i>	Pointer to the $n*n$ dense matrix
<i>x</i>	Pointer to the REAL array with length $n-1$
<i>y</i>	Pointer to the REAL array with length $n-1$
<i>n</i>	the dimension of the dense matrix

Author

Xiaozhe Hu

Date

2010/10/25

Note

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

Definition at line 1480 of file blas_smat.c.

9.14.2.28 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 saturation part $1*1$.

Parameters

<i>A</i>	Pointer to the 2*2 dense matrix
<i>x</i>	Pointer to the REAL array with length 1
<i>y</i>	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 explicitly use saturation block!!

Definition at line 1356 of file blas_smat.c.

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

Compute $ys := ys - Ass \cdot xs$, where 'A' is a 3*3 dense matrix, Ass is its saturation part 2*2.

Parameters

<i>A</i>	Pointer to the 3*3 dense matrix
<i>x</i>	Pointer to the REAL array with length 2
<i>y</i>	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 explicitly use saturation block!!

Definition at line 1380 of file blas_smat.c.

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

Compute $ys := ys - Ass \cdot xs$, where 'A' is a 5*5 dense matrix, Ass is its saturation part 4*4.

Parameters

<i>A</i>	Pointer to the 5*5 dense matrix
<i>x</i>	Pointer to the REAL array with length 4
<i>y</i>	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 explicitly use saturation block!!

Definition at line 1408 of file blas_smat.c.

9.14.2.31 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 saturation part 6*6.

Parameters

<i>A</i>	Pointer to the 7*7 dense matrix
<i>x</i>	Pointer to the REAL array with length 6
<i>y</i>	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 explicitly use saturation block!!

Definition at line 1442 of file blas_smat.c.

9.14.2.32 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

<i>A</i>	Pointer to the n*n dense matrix
<i>x</i>	Pointer to the REAL array with length n
<i>y</i>	Pointer to the REAL array with length n
<i>n</i>	Dimension of the dense matrix

Author

Zhiyang Zhou

Date

2010/10/25

Definition at line 974 of file blas_smat.c.

9.14.2.33 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

<i>A</i>	Pointer to the 3*3 dense matrix
<i>x</i>	Pointer to the REAL array with length 3
<i>y</i>	Pointer to the REAL array with length 3

Author

Xiaozhe Hu

Date

2011/11/18

Definition at line 855 of file blas_smat.c.

9.14.2.34 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

<i>A</i>	Pointer to the 3*3 dense matrix
<i>x</i>	Pointer to the REAL array with length 3
<i>y</i>	Pointer to the REAL array with length 3

Author

Zhiyang Zhou, Xiaozhe Hu

Date

2010/10/25

Definition at line 881 of file blas_smat.c.

9.14.2.35 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 908 of file blas_smat.c.

9.14.2.36 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

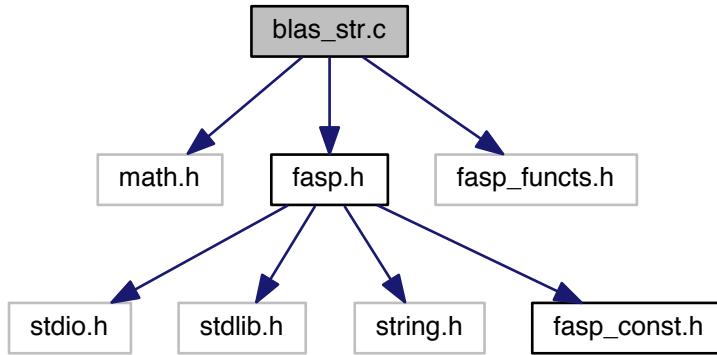
Definition at line 939 of file blas_smat.c.

9.15 blas_str.c File Reference

BLAS operations for [dSTRmat](#) matrices.

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

Include dependency graph for blas_str.c:



Functions

- `void fasp_blas_dstr_aApxy (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)`
 $B = D^{-1}A$.

9.15.1 Detailed Description

BLAS operations for `dSTRmat` matrices.

9.15.2 Function Documentation

9.15.2.1 void fasp_blas_dstr_aApxy (REAL alpha, dSTRmat * A, REAL * x, REAL * y)

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

Parameters

<code>alpha</code>	REAL factor alpha
<code>A</code>	Pointer to <code>dSTRmat</code> matrix
<code>x</code>	Pointer to REAL array
<code>y</code>	Pointer to REAL array

Author

Zhiyang Zhou, Xiaozhe Hu, Shiquan Zhang

Date

2010/10/15

Definition at line 47 of file blas_str.c.

9.15.2.2 void fasp_bla_dstr_mxv (dSTRmat * A, REAL * x, REAL * y)

Matrix-vector multiplication $y = A \cdot x$.

Parameters

<i>A</i>	Pointer to <code>dSTRmat</code> matrix
<i>x</i>	Pointer to <code>REAL</code> array
<i>y</i>	Pointer to <code>REAL</code> array

Author

Chensong Zhang

Date

04/27/2013

Definition at line 117 of file blas_str.c.

Here is the call graph for this function:



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

$B = D^{-1}A$.

Parameters

<i>A</i>	Pointer to a ' <code>dSTRmat</code> ' type matrix A
<i>B</i>	Pointer to a ' <code>dSTRmat</code> ' type matrix B

Author

Shiquan Zhang

Date

2010/10/15

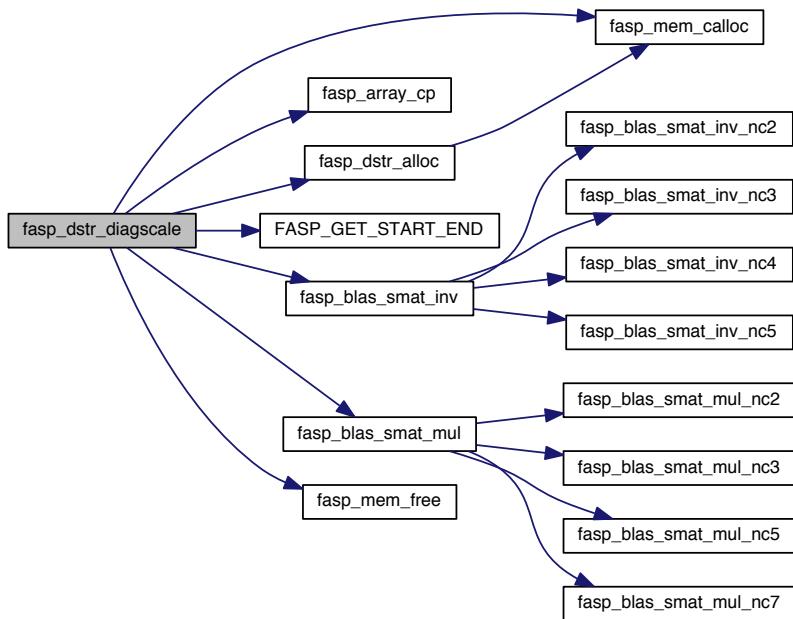
Modified by Chunsheng Feng, Zheng Li

Date

08/30/2012

Definition at line 142 of file blas_str.c.

Here is the call graph for this function:

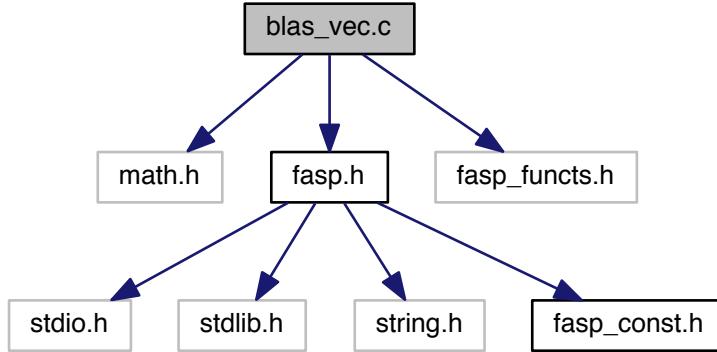


9.16 blas_vec.c File Reference

BLAS operations for vectors.

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

Include dependency graph for blas_vec.c:



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.

9.16.1 Detailed Description

BLAS operations for vectors.

9.16.2 Function Documentation

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

$y = a*x + y$

Parameters

<i>a</i>	REAL factor a
<i>x</i>	Pointer to dvector x
<i>y</i>	Pointer to dvector y

Author

Chensong Zhang

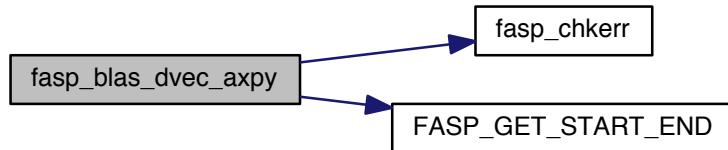
Date

07/01/209

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

Definition at line 33 of file blas_vec.c.

Here is the call graph for this function:



9.16.2.2 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

<i>a</i>	REAL factor a
<i>x</i>	Pointer to dvector x
<i>y</i>	Pointer to dvector y
<i>z</i>	Pointer to dvector z

Author

Chensong Zhang

Date

07/01/209

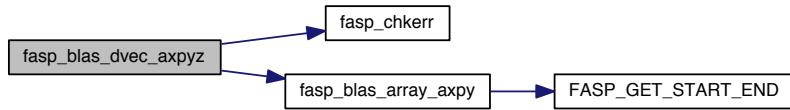
Modified by Chunsheng Feng, Xiaoqiang Yue

Date

05/23/2012

Definition at line 85 of file blas_vec.c.

Here is the call graph for this function:



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

Inner product of two vectors (x,y)

Parameters

<i>x</i>	Pointer to dvector x
<i>y</i>	Pointer to dvector y

Returns

Inner product

Author

Chensong Zhang

Date

07/01/2009

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

05/23/2012

Definition at line 121 of file blas_vec.c.

9.16.2.4 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

07/01/2009

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

05/23/2012

Definition at line 222 of file blas_vec.c.

9.16.2.5 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

07/01/2009

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

05/23/2012

Definition at line 265 of file blas_vec.c.

9.16.2.6 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

07/01/2009

Definition at line 305 of file blas_vec.c.

9.16.2.7 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

07/01/2009

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

05/23/2012

Definition at line 167 of file blas_vec.c.

Here is the call graph for this function:

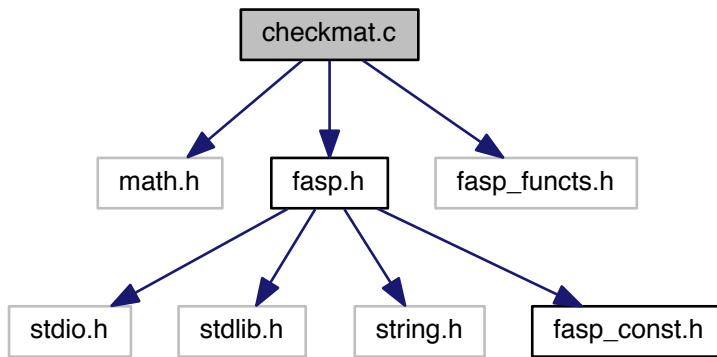


9.17 checkmat.c File Reference

Check matrix properties.

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

Include dependency graph for checkmat.c:



Functions

- [INT fasp_check_diagpos \(dCSRmat *A\)](#)
Check positivity of diagonal entries of a CSR sparse matrix.
- [SHORT fasp_check_diagzero \(dCSRmat *A\)](#)
Check whether 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.

9.17.1 Detailed Description

Check matrix properties.

9.17.2 Function Documentation

9.17.2.1 **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`.

Here is the call graph for this function:



9.17.2.2 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 checks 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

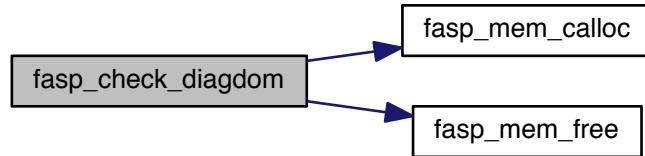
Shuo Zhang

Date

03/29/2009

Definition at line 108 of file checkmat.c.

Here is the call graph for this function:



9.17.2.3 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

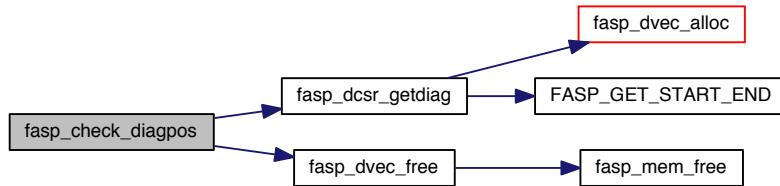
Shuo Zhang

Date

03/29/2009

Definition at line 27 of file checkmat.c.

Here is the call graph for this function:



9.17.2.4 SHORT fasp_check_diagzero(dCSRmat * A)

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

Parameters

A	pointer to the dCSRmat matrix
---	---

Returns

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

Author

Shuo Zhang

Date

03/29/2009

Definition at line 64 of file checkmat.c.

9.17.2.5 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.

Here is the call graph for this function:

**9.17.2.6 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

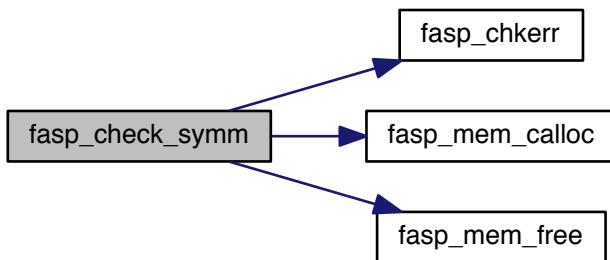
Shuo Zhang

Date

03/29/2009

Definition at line 153 of file `checkmat.c`.

Here is the call graph for this function:

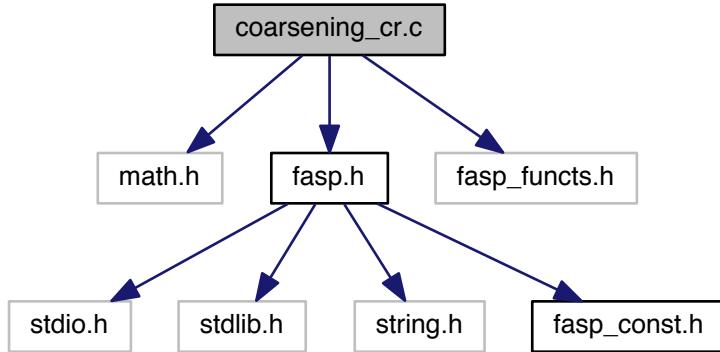


9.18 coarsening_cr.c File Reference

Coarsening with Brannick-Falgout strategy.

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

Include dependency graph for coarsening_cr.c:



Functions

- `INT fasp_amg_coarsening_cr (INT i_0, INT i_n, dCSRmat *A, ivec &vertices, AMG_param ¶m)`
CR coarsening.

9.18.1 Detailed Description

Coarsening with Brannick-Falgout strategy.

9.18.2 Function Documentation

9.18.2.1 INT fasp_amg_coarsening_cr (INT i_0, INT i_n, dCSRmat * A, ivec * vertices, AMG_param * param)

CR coarsening.

Parameters

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

Author

James Brannick

Date

04/21/2010

Modified by Chunsheng Feng, Zheng Li

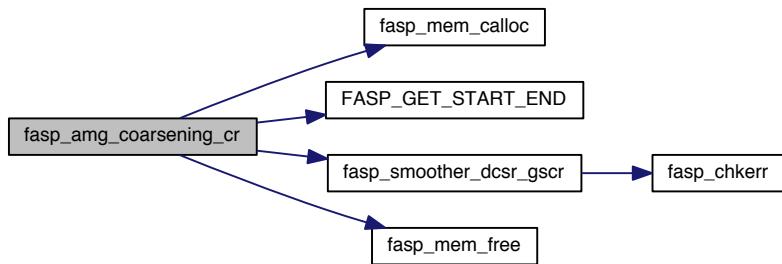
Date

10/14/2012

CR STAGES

Definition at line 41 of file coarsening_cr.c.

Here is the call graph for this function:

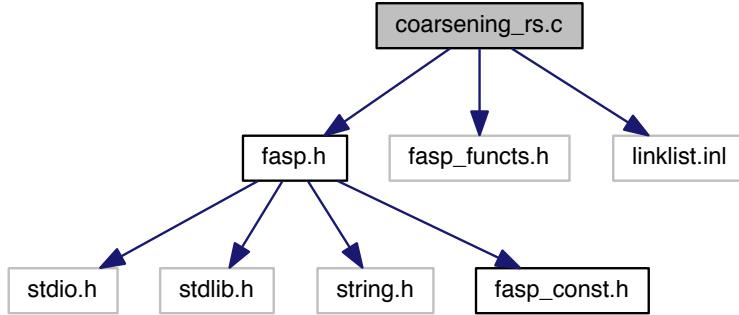


9.19 coarsening_rs.c File Reference

Coarsening with a modified Ruge-Stuben strategy.

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

Include dependency graph for coarsening_rs.c:



Functions

- **SHORT fasp_amg_coarsening_rs (dCSRmat *A, ivecotor *vertices, dCSRmat *P, iCSRmat *S, AMG_param *param)**

Standard and aggressive coarsening schemes.

9.19.1 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!!!

9.19.2 Function Documentation

9.19.2.1 SHORT fasp_amg_coarsening_rs (dCSRmat * A, ivecotor * 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
<i>param</i>	Pointer to AMG_param : AMG parameters

Returns

FASP_SUCCESS or error message

Author

Xuehai Huang, Chensong Zhang, Xiaozhe Hu, Ludmil Zikatanov

Date

09/06/2010

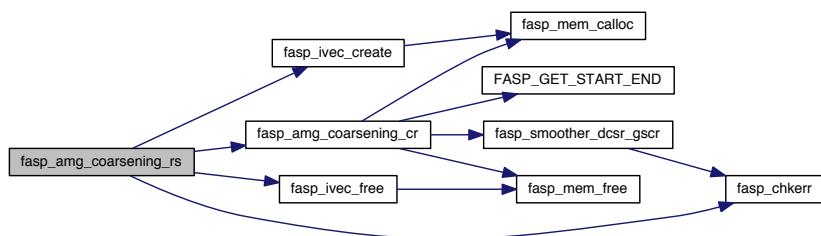
Note

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.

Here is the call graph for this function:

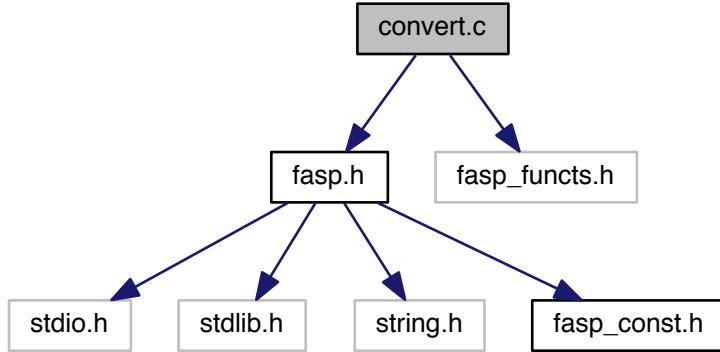


9.20 convert.c File Reference

Some utilities for format conversion.

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

Include dependency graph for convert.c:



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_bbyteTodouble (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, INT vlength, INT endianflag)`
Swap order of a REAL number.

9.20.1 Detailed Description

Some utilities for format conversion.

9.20.2 Function Documentation

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

Swap order of an INT number.

Parameters

<i>inum</i>	An INT value
<i>ilength</i>	Length of INT: 2 for short, 4 for int, 8 for long
<i>endianflag</i>	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.

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

Swap order of a REAL number.

Parameters

<i>rnum</i>	An REAL value
<i>ilength</i>	Length of INT: 2 for short, 4 for int, 8 for long
<i>endianflag</i>	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.

9.20.2.3 double fasp_aux_bbyteTodouble (unsigned char bytes[])

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

Parameters

<i>bytes</i>	A unsigned char
--------------	-----------------

Returns

Unsigend long ineger after swapping

Author

Chensong Zhang

Date

11/16/2009

Definition at line 74 of file convert.c.

9.20.2.4 unsigned long fasp_aux_change_endian4(unsigned long x)

Swap order for different endian systems.

Parameters

<i>x</i>	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.

9.20.2.5 double fasp_aux_change_endian8(double x)

Swap order for different endian systems.

Parameters

<i>x</i>	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.

9.21 doxygen.h File Reference

Main page for Doygen documentation.

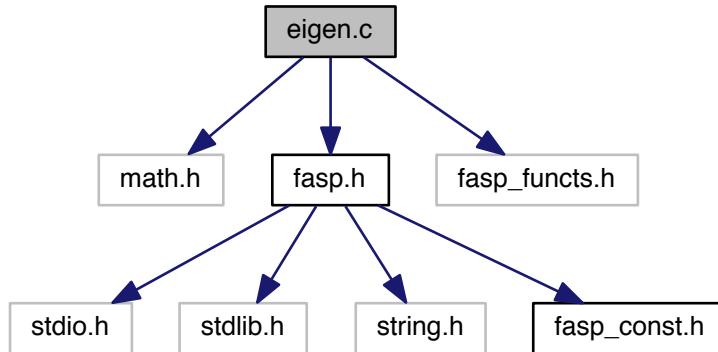
9.21.1 Detailed Description

Main page for Doygen documentation.

9.22 eigen.c File Reference

Simple subroutines for compute the extreme eigenvalues.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
Include dependency graph for eigen.c:
```



Functions

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

9.22.1 Detailed Description

Simple subroutines for compute the extreme eigenvalues.

9.22.2 Function Documentation

9.22.2.1 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 <code>dCSRmat</code> matrix
<i>tol</i>	Tolerance for stopping the power method
<i>maxit</i>	Max number of iterations

Returns

Largest eigenvalue

Author

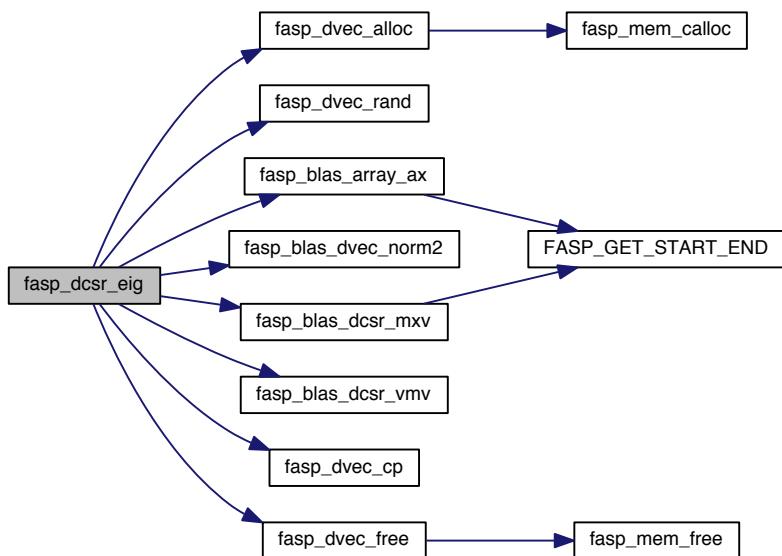
Xiaozhe Hu

Date

01/25/2011

Definition at line 29 of file eigen.c.

Here is the call graph for this function:



9.23 factor.f File Reference

LU factorization for CSR matrix.

Functions/Subroutines

- subroutine **sfactr** (ia, ja, n, iu, ju, ip, nwku)
- subroutine **sfactr_new** (ia, ja, n, iu, ju, ip, nwku, mem_chk)
- subroutine **factor** (ia, ja, n, iu, ju, ip, iup, an, ad, un, di)
- subroutine **forbac** (iu, ju, un, di, n, x)

9.23.1 Detailed Description

LU factorization for CSR matrix.

Author

Ludmil Zikatanov

Date

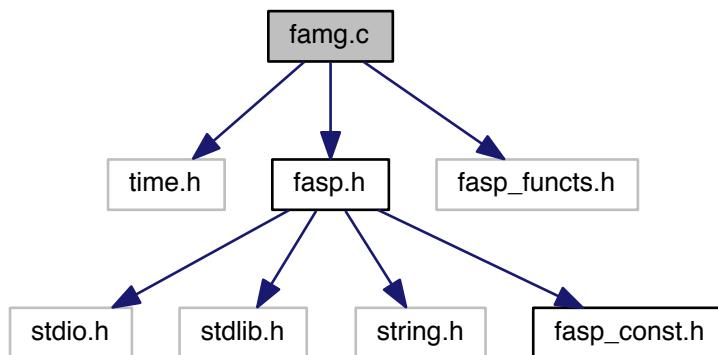
01/01/2002

9.24 famg.c File Reference

full AMG method as an iterative solver (main file)

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

Include dependency graph for famg.c:



Functions

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

Solve Ax=b by full AMG.

9.24.1 Detailed Description

full AMG method as an iterative solver (main file)

9.24.2 Function Documentation

9.24.2.1 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

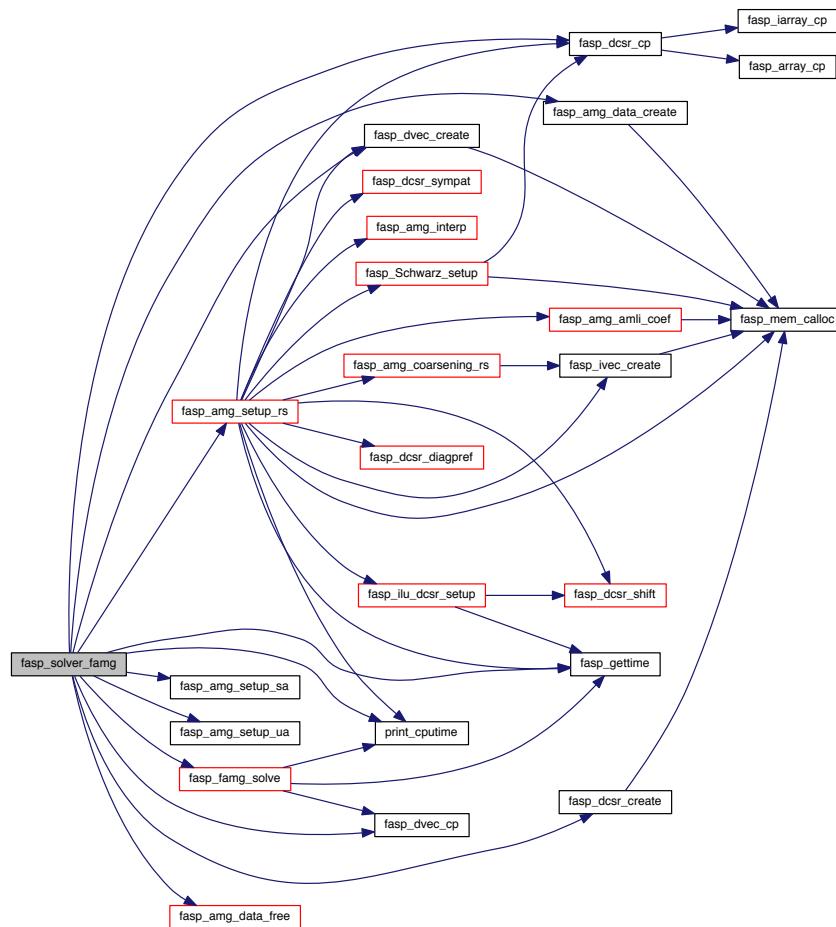
Date

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.

Here is the call graph for this function:

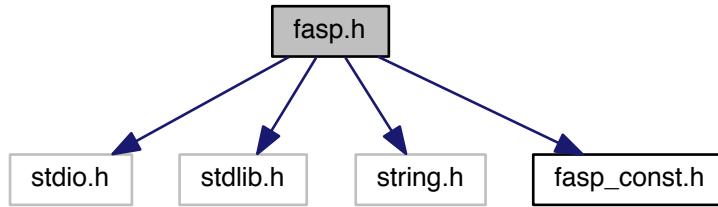


9.25 fasp.h File Reference

Main header file for FASPI.

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

Include dependency graph for fasp.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 [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_USE_ILU](#) ON
For external software package support.
- #define [DLMalloc](#) OFF
- #define [NEDMalloc](#) OFF
- #define [RS_C1](#) ON
Flags for internal uses (change with caution!!!)
- #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 **PRT_INT**(A) printf("### DEBUG: %s = %d\n", #A, (A))
Definition of print command in DEBUG mode.
- #define **PRT_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**

9.25.1 Detailed Description

Main header file for FASP.

This header file contains general constants and data structures used in FASP.

Note

Only define macros and data structures, no function decorations.

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

9.25.2 Macro Definition Documentation

9.25.2.1 #define __FASP_HEADER__

indicate [fasp.h](#) has been included before

Definition at line 28 of file fasp.h.

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

absolute value of a

Definition at line 65 of file fasp.h.

9.25.2.3 #define DIAGONAL_PREF OFF

order each row such that diagonal appears first

Definition at line 47 of file fasp.h.

9.25.2.4 #define DLMALLOC OFF

use dlmalloc instead of standard malloc

Definition at line 38 of file fasp.h.

9.25.2.5 #define FASP_GSRB 1

MG level 0 use RedBlack Gauss Seidel Smoothing

Definition at line 1157 of file fasp.h.

9.25.2.6 #define FASP_USE_ILU ON

For external software package support.

enable ILU or not

Definition at line 37 of file fasp.h.

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

is a \geq b?

Definition at line 71 of file fasp.h.

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

Definition of >, \geq , <, \leq , and isnan.

is a $>$ b?

Definition at line 70 of file fasp.h.

9.25.2.9 #define INT int

regular integer type: int or long

Definition at line 55 of file fasp.h.

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

is a == NAN?

Definition at line 74 of file fasp.h.

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

is a <= b?

Definition at line 73 of file fasp.h.

9.25.2.12 #define LONG long

long integer type

Definition at line 56 of file fasp.h.

9.25.2.13 #define LONGLONG long long

long integer type

Definition at line 57 of file fasp.h.

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

is a < b?

Definition at line 72 of file fasp.h.

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

Definition of max, min, abs.

bigger one in a and b

Definition at line 63 of file fasp.h.

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

smaller one in a and b

Definition at line 64 of file fasp.h.

9.25.2.17 #define NEDMALLOC OFF

use nedmalloc instead of standard malloc

Definition at line 39 of file fasp.h.

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

Definition of print command in DEBUG mode.

print an integer

Definition at line 79 of file fasp.h.

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

print a real num

Definition at line 80 of file fasp.h.

9.25.2.20 #define REAL double

float type

Definition at line 58 of file fasp.h.

9.25.2.21 #define RS_C1 ON

Flags for internal uses (change with caution!!!)

CF splitting of RS: check C1 Criterion

Definition at line 44 of file fasp.h.

9.25.2.22 #define SHORT short

FASP integer and floating point numbers.

short integer type

Definition at line 54 of file fasp.h.

9.25.3 Typedef Documentation

9.25.3.1 typedef struct dCOOmat dCOOmat

Sparse matrix of REAL type in COO format

9.25.3.2 typedef struct dCSRLmat dCSRLmat

Sparse matrix of REAL type in CSRL format

9.25.3.3 `typedef struct dCSRmat dCSRmat`

Sparse matrix of REAL type in CSR format

9.25.3.4 `typedef struct ddemat ddemat`

Dense matrix of REAL type

9.25.3.5 `typedef struct dSTRmat dSTRmat`

Structured matrix of REAL type

9.25.3.6 `typedef struct dvector dvector`

Vector of REAL type

9.25.3.7 `typedef struct grid2d grid2d`

2D grid type for plotting

9.25.3.8 `typedef struct iCOOmat iCOOmat`

Sparse matrix of INT type in COO format

9.25.3.9 `typedef struct iCSRmat iCSRmat`

Sparse matrix of INT type in CSR format

9.25.3.10 `typedef struct idenmat idenmat`

Dense matrix of INT type

9.25.3.11 `typedef struct ivector ivector`

Vector of INT type

9.25.3.12 `typedef ListElement* LinkList`

List of linkslinked list

Definition at line 1152 of file fasp.h.

9.25.3.13 `typedef struct linked_list ListElement`

Linked element in list

9.25.3.14 **typedef const grid2d* pcgrid2d**

Grid in 2d

Definition at line 1106 of file fasp.h.

9.25.3.15 **typedef grid2d* pgrid2d**

Grid in 2d

Definition at line 1104 of file fasp.h.

9.25.4 Variable Documentation

9.25.4.1 **INT count**

Counter for multiple calls

9.25.4.2 **INT* IMAP**

Red Black Gs Smoother imap

9.25.4.3 **INT MAXIMAP**

Red Black Gs Smoother max DOFs of reservoir

9.25.4.4 **INT nx_rb**

Red Black Gs Smoother Nx

9.25.4.5 **INT ny_rb**

Red Black Gs Smoother Ny

9.25.4.6 **INT nz_rb**

Red Black Gs Smoother Nz

9.25.4.7 **unsigned INT total_alloc_count**

Total allocated memory amount.

total allocation times

Definition at line 33 of file memory.c.

9.25.4.8 **unsigned INT total_alloc_mem**

total allocated memory

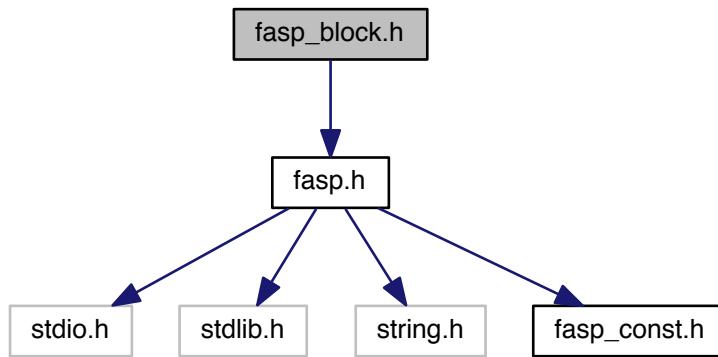
Definition at line 32 of file `memory.c`.

9.26 fasp_block.h File Reference

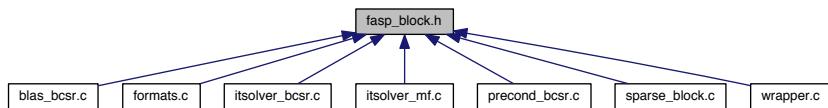
Main header file for FASP (block matrices)

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

Include dependency graph for `fasp_block.h`:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `dBSRmat`
Block sparse row storage matrix of REAL type.
- struct `block_dCSRmat`
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 diagonal preconditioner for [dBSRmat](#) matrices.
- struct [precond_data_bsr](#)
Data passed to the preconditioners.
- struct [precond_block_reservoir_data](#)
Data passed to the preconditioner for preconditioning reservoir simulation problems.
- struct [precond_block_data](#)
Data passed to the preconditioner for block preconditioning for [block_dCSRmat](#) 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.

Typedefs

- [typedef struct dBSRmat dBSRmat](#)
- [typedef struct block_dCSRmat block_dCSRmat](#)
- [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](#)

9.26.1 Detailed Description

Main 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 decorations.

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.

9.26.2 Typedef Documentation

9.26.2.1 `typedef struct block_BSR block_BSR`

Block of BSR matrices of REAL type

9.26.2.2 **typedef struct block_dCSRmat block_dCSRmat**

Matrix of REAL type in Block CSR format

9.26.2.3 **typedef struct block_dvector block_dvector**

Vector of REAL type in Block format

9.26.2.4 **typedef struct block_iCSRmat block_iCSRmat**

Matrix of INT type in Block CSR format

9.26.2.5 **typedef struct block_ivector block_ivector**

Vector of INT type in Block format

9.26.2.6 **typedef struct block_Reservoir block_Reservoir**

Special block matrix for Reservoir Simulation

9.26.2.7 **typedef struct dBSRmat dBSRmat**

Matrix of REAL type in BSR format

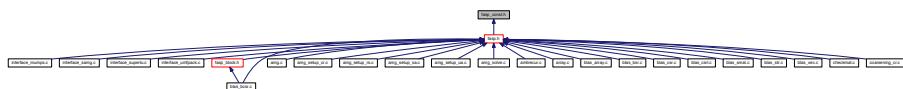
9.26.2.8 `typedef struct precond_block_reservoir_data precond_block_reservoir_data`

Precond data for Reservoir Simulation

9.27 fasp_const.h File Reference

Definition of all kinds of messages, including error messages, solver types, etc.

This graph shows which files directly or indirectly include this file:



Macros

- #define **BIGREAL** 1e+20

Some global constants.
- #define **SMALLREAL** 1e-20
- #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 **STAG_RATIO** 1e-4
- #define **MAX_STAG** 20
- #define **MAX_RESTART** 20
- #define **OPENMP HOLDS** 2000
- #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_bCSR 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_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 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 **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 **SMOOTHER_BLKOIL** 11
- Definition of specialized smoother types.*
- #define **SMOOTHER_SPETEN** 19
- #define **COARSE_RS** 1
- Definition of coarsening types.*
- #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 **USERDEFINED** 0
- Type of ordering for smoothers.*

- #define CPFIRST 1
- #define FPFIRST -1
- #define ASCEND 12
- #define DESCEND 21

9.27.1 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 -> SMOOTHING_CG. Modified by Chensong Zhang on 02/16/2013: GS -> SMOOTHING_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.

9.27.2 Macro Definition Documentation

9.27.2.1 #define AMLI_CYCLE 3

AMLI-cycle

Definition at line 184 of file fasp_const.h.

9.27.2.2 #define ASCEND 12

Ascending order

Definition at line 243 of file fasp_const.h.

9.27.2.3 #define BIGREAL 1e+20

Some global constants.

A large real number

Definition at line 27 of file fasp_const.h.

9.27.2.4 #define CF_ORDER 1

C/F order smoothing

Definition at line 235 of file fasp_const.h.

9.27.2.5 #define CGPT 1

Coarse grid points

Definition at line 228 of file fasp_const.h.

9.27.2.6 #define CLASSIC_AMG 1

Definition of AMG types.

classic AMG

Definition at line 169 of file fasp_const.h.

9.27.2.7 #define COARSE_AC 4

Aggressive coarsening

Definition at line 212 of file fasp_const.h.

9.27.2.8 #define COARSE_CR 3

Compatible relaxation

Definition at line 211 of file fasp_const.h.

9.27.2.9 #define COARSE_MIS 5

Aggressive coarsening based on MIS

Definition at line 213 of file fasp_const.h.

9.27.2.10 #define COARSE_RS 1

Definition of coarsening types.

Classical coarsening

Definition at line 210 of file fasp_const.h.

9.27.2.11 #define CPFIRST 1

C-points first order

Definition at line 241 of file fasp_const.h.

9.27.2.12 #define DESCEND 21

Descending order

Definition at line 244 of file fasp_const.h.

9.27.2.13 #define ERROR_ALLOC_MEM -20

fail to allocate memory

Definition at line 52 of file fasp_const.h.

9.27.2.14 #define ERROR_AMG_COARSE_TYPE -32

unknown coarsening type

Definition at line 59 of file fasp_const.h.

9.27.2.15 #define ERROR_AMG_COARSEING -33

coarsening step failed to complete

Definition at line 60 of file fasp_const.h.

9.27.2.16 #define ERROR_AMG_INTERP_TYPE -30

unknown interpolation type

Definition at line 57 of file fasp_const.h.

9.27.2.17 #define ERROR_AMG_SMOOTH_TYPE -31

unknown smoother type

Definition at line 58 of file fasp_const.h.

9.27.2.18 #define ERROR_DATA_STRUCTURE -21

problem with data structures

Definition at line 53 of file fasp_const.h.

9.27.2.19 #define ERROR_DATA_ZERODIAG -22

matrix has zero diagonal entries

Definition at line 54 of file fasp_const.h.

9.27.2.20 #define ERROR_DUMMY_VAR -23

unexpected input data

Definition at line 55 of file fasp_const.h.

9.27.2.21 #define ERROR_INPUT_PAR -13

wrong input argument

Definition at line 46 of file fasp_const.h.

9.27.2.22 #define ERROR_LIC_TYPE -80

wrong license type

Definition at line 75 of file fasp_const.h.

9.27.2.23 #define ERROR_MAT_SIZE -15

wrong problem size

Definition at line 48 of file fasp_const.h.

9.27.2.24 #define ERROR_MISC -19

other error

Definition at line 50 of file fasp_const.h.

9.27.2.25 #define ERROR_NUM_BLOCKS -18

wrong number of blocks

Definition at line 49 of file fasp_const.h.

9.27.2.26 #define ERROR_OPEN_FILE -10

fail to open a file

Definition at line 44 of file fasp_const.h.

9.27.2.27 #define ERROR_QUAD_DIM -61

unsupported quadrature dim

Definition at line 73 of file fasp_const.h.

9.27.2.28 #define ERROR_QUAD_TYPE -60

unknown quadrature type

Definition at line 72 of file fasp_const.h.

9.27.2.29 #define ERROR_REGRESS -14

regression test fail

Definition at line 47 of file fasp_const.h.

9.27.2.30 #define ERROR_SOLVER_EXIT -49

solver does not quit successfully

Definition at line 70 of file fasp_const.h.

9.27.2.31 #define ERROR_SOLVER_ILUSETUP -45

ILU setup error

Definition at line 67 of file fasp_const.h.

9.27.2.32 #define ERROR_SOLVER_MAXIT -48

maximal iteration number exceeded

Definition at line 69 of file fasp_const.h.

9.27.2.33 #define ERROR_SOLVER_MISC -46

misc solver error during run time

Definition at line 68 of file fasp_const.h.

9.27.2.34 #define ERROR_SOLVER_PRECTYPE -41

unknown precond type

Definition at line 63 of file fasp_const.h.

9.27.2.35 #define ERROR_SOLVER_SOLSTAG -43

solver's solution is too small

Definition at line 65 of file fasp_const.h.

9.27.2.36 #define ERROR_SOLVER_STAG -42

solver stagnates

Definition at line 64 of file fasp_const.h.

9.27.2.37 #define ERROR_SOLVER_TOLSMALL -44

solver's tolerance is too small

Definition at line 66 of file fasp_const.h.

9.27.2.38 #define ERROR_SOLVER_TYPE -40

unknown solver type

Definition at line 62 of file fasp_const.h.

9.27.2.39 #define ERROR_UNKNOWN -99

an unknown error type

Definition at line 77 of file fasp_const.h.

9.27.2.40 #define ERROR_WRONG_FILE -11

input contains wrong format

Definition at line 45 of file fasp_const.h.

9.27.2.41 #define FALSE 0

logic FALSE

Definition at line 83 of file fasp_const.h.

9.27.2.42 #define FASP_SUCCESS 0

Definition of return status and error messages.

return from function successfully

Definition at line 42 of file fasp_const.h.

9.27.2.43 #define FGPT 0

Fine grid points

Definition at line 227 of file fasp_const.h.

9.27.2.44 #define FPFIRST -1

F-points first order

Definition at line 242 of file fasp_const.h.

9.27.2.45 #define G0PT -5

Type of vertices (DOFs) for coarsening.

Cannot fit in aggregates

Definition at line 225 of file fasp_const.h.

9.27.2.46 #define ILUk 1

Type of ILU methods.

ILUk

Definition at line 162 of file fasp_const.h.

9.27.2.47 #define ILUT 2

ILUT

Definition at line 163 of file fasp_const.h.

9.27.2.48 #define ILUtp 3

ILUtp

Definition at line 164 of file fasp_const.h.

9.27.2.49 #define INTERP_DIR 1

Definition of interpolation types.

Direct interpolation

Definition at line 218 of file fasp_const.h.

9.27.2.50 #define INTERP_ENG 3

energy minimization interpolation

Definition at line 220 of file fasp_const.h.

9.27.2.51 #define INTERP_STD 2

Standard interpolation

Definition at line 219 of file fasp_const.h.

9.27.2.52 #define ISPT 2

Isolated points

Definition at line 229 of file fasp_const.h.

9.27.2.53 #define MAT_bBSR 5

block matrix of BSR for bordered systems

Definition at line 109 of file fasp_const.h.

9.27.2.54 #define MAT_bCSR 4

block matrix of CSR

Definition at line 108 of file fasp_const.h.

9.27.2.55 #define MAT_BSR 2

block-wise compressed sparse row

Definition at line 106 of file fasp_const.h.

9.27.2.56 #define MAT_CSR 1

compressed sparse row

Definition at line 105 of file fasp_const.h.

9.27.2.57 #define MAT_CSRL 6

modified CSR to reduce cache missing
Definition at line 110 of file fasp_const.h.

9.27.2.58 #define MAT_FREE 0

Definition of matrix format.
matrix-free format: only mxv action
Definition at line 104 of file fasp_const.h.

9.27.2.59 #define MAT_STR 3

structured sparse matrix
Definition at line 107 of file fasp_const.h.

9.27.2.60 #define MAT_SymCSR 7

symmetric CSR format
Definition at line 111 of file fasp_const.h.

9.27.2.61 #define MAX_AMG_LVL 20

Maximal AMG coarsening level
Definition at line 30 of file fasp_const.h.

9.27.2.62 #define MAX_CRATE 20.0

Maximal coarsening ratio
Definition at line 33 of file fasp_const.h.

9.27.2.63 #define MAX_REFINE_LVL 20

Maximal refinement level
Definition at line 29 of file fasp_const.h.

9.27.2.64 #define MAX_RESTART 20

Maximal number of restarting for BiCGStab
Definition at line 36 of file fasp_const.h.

9.27.2.65 #define MAX_STAG 20

Maximal number of stagnation times

Definition at line 35 of file fasp_const.h.

9.27.2.66 #define MIN_CDOF 20

Minimal number of coarsest variables

Definition at line 31 of file fasp_const.h.

9.27.2.67 #define MIN_CRATE 0.9

Minimal coarsening ratio

Definition at line 32 of file fasp_const.h.

9.27.2.68 #define NL_AMLI_CYCLE 4

Nonlinear AMLI-cycle

Definition at line 185 of file fasp_const.h.

9.27.2.69 #define NO_ORDER 0

Definition of smoothing order.

Natural order smoothing

Definition at line 234 of file fasp_const.h.

9.27.2.70 #define OFF 0

turn off certain parameter

Definition at line 89 of file fasp_const.h.

9.27.2.71 #define ON 1

Definition of switch.

turn on certain parameter

Definition at line 88 of file fasp_const.h.

9.27.2.72 #define OPENMP_HOLDS 2000

Switch to sequence version when size is small

Definition at line 37 of file fasp_const.h.

9.27.2.73 #define PAIRWISE 1

Definition of aggregation types.

pairwise aggregation

Definition at line 176 of file fasp_const.h.

9.27.2.74 #define PREC_AMG 2

with AMG precond

Definition at line 154 of file fasp_const.h.

9.27.2.75 #define PREC_DIAG 1

with diagonal precond

Definition at line 153 of file fasp_const.h.

9.27.2.76 #define PREC_FMG 3

with full AMG precond

Definition at line 155 of file fasp_const.h.

9.27.2.77 #define PREC_ILU 4

with ILU precond

Definition at line 156 of file fasp_const.h.

9.27.2.78 #define PREC_NULL 0

Definition of preconditioner type for iterative methods.

with no precond

Definition at line 152 of file fasp_const.h.

9.27.2.79 #define PREC_SCHWARZ 5

with Schwarz preconditioner

Definition at line 157 of file fasp_const.h.

9.27.2.80 #define PRINT_ALL 10

everything: all printouts, including files

Definition at line 99 of file fasp_const.h.

9.27.2.81 #define PRINT_MIN 1

quiet: min info, error, important warnings

Definition at line 95 of file fasp_const.h.

9.27.2.82 #define PRINT_MORE 4

more: print some useful debug information

Definition at line 97 of file fasp_const.h.

9.27.2.83 #define PRINT_MOST 8

most: maximal printouts, no files

Definition at line 98 of file fasp_const.h.

9.27.2.84 #define PRINT_NONE 0

Print level for all subroutines – not including DEBUG output.

silent: no printout at all

Definition at line 94 of file fasp_const.h.

9.27.2.85 #define PRINT_SOME 2

some: more info, less important warnings

Definition at line 96 of file fasp_const.h.

9.27.2.86 #define SA_AMG 2

smoothed aggregation AMG

Definition at line 170 of file fasp_const.h.

9.27.2.87 #define SMALLREAL 1e-20

A small real number

Definition at line 28 of file fasp_const.h.

9.27.2.88 #define SMOOTHER_BLKOIL 11

Definition of specialized smoother types.

Used in monolithic AMG for black-oil

Definition at line 204 of file fasp_const.h.

9.27.2.89 #define SMOOTH_CG 4

CG as a smoother

Definition at line 193 of file fasp_const.h.

9.27.2.90 #define SMOOTH_GS 2

Gauss-Seidel smoother

Definition at line 191 of file fasp_const.h.

9.27.2.91 #define SMOOTH_GSOR 7

GS + SOR smoother

Definition at line 196 of file fasp_const.h.

9.27.2.92 #define SMOOTH_JACOBI 1

Definition of standard smoother types.

Jacobi smoother

Definition at line 190 of file fasp_const.h.

9.27.2.93 #define SMOOTH_L1DIAG 10

L1 norm diagonal scaling smoother

Definition at line 199 of file fasp_const.h.

9.27.2.94 #define SMOOTH_POLY 9

Polynomial smoother

Definition at line 198 of file fasp_const.h.

9.27.2.95 #define SMOOTH_SGS 3

Symmetric Gauss-Seidel smoother

Definition at line 192 of file fasp_const.h.

9.27.2.96 #define SMOOTH_SGSOR 8

SGS + SSOR smoother

Definition at line 197 of file fasp_const.h.

9.27.2.97 #define SMOOTH_SOR 5

SOR smoother

Definition at line 194 of file fasp_const.h.

9.27.2.98 #define SMOOTH_SPETEN 19

Used in monolithic AMG for black-oil

Definition at line 205 of file fasp_const.h.

9.27.2.99 #define SMOOTH_SSOR 6

SSOR smoother

Definition at line 195 of file fasp_const.h.

9.27.2.100 #define SOLVER_AMG 21

AMG as an iterative solver

Definition at line 135 of file fasp_const.h.

9.27.2.101 #define SOLVER_BiCGstab 2

Bi-Conjugate Gradient Stabilized

Definition at line 119 of file fasp_const.h.

9.27.2.102 #define SOLVER_CG 1

Conjugate Gradient

Definition at line 118 of file fasp_const.h.

9.27.2.103 #define SOLVER_DEFAULT 0

Definition of solver types for iterative methods.

Use default solver in FASP

Definition at line 116 of file fasp_const.h.

9.27.2.104 #define SOLVER_FMGI 22

Full AMG as an solver

Definition at line 136 of file fasp_const.h.

9.27.2.105 #define SOLVER_GCG 7

Generalized Conjugate Gradient

Definition at line 124 of file fasp_const.h.

9.27.2.106 #define SOLVER_GCR 8

Generalized Conjugate Residual

Definition at line 125 of file fasp_const.h.

9.27.2.107 #define SOLVER_GMRES 4

Generalized Minimal Residual

Definition at line 121 of file fasp_const.h.

9.27.2.108 #define SOLVER_MinRes 3

Minimal Residual

Definition at line 120 of file fasp_const.h.

9.27.2.109 #define SOLVER_MUMPS 33

MUMPS Direct Solver

Definition at line 140 of file fasp_const.h.

9.27.2.110 #define SOLVER_SBiCGstab 12

BiCGstab with safe net

Definition at line 128 of file fasp_const.h.

9.27.2.111 #define SOLVER_SCG 11

Conjugate Gradient with safe net

Definition at line 127 of file fasp_const.h.

9.27.2.112 #define SOLVER_SGCG 17

GCG with safe net

Definition at line 133 of file fasp_const.h.

9.27.2.113 #define SOLVER_SGMRES 14

GMRes with safe net

Definition at line 130 of file fasp_const.h.

9.27.2.114 #define SOLVER_SMinRes 13

MinRes with safe net

Definition at line 129 of file fasp_const.h.

9.27.2.115 #define SOLVER_SUPERLU 31

SuperLU Direct Solver

Definition at line 138 of file fasp_const.h.

9.27.2.116 #define SOLVER_SVFGMRES 16

Variable-restart FGMRES with safe net

Definition at line 132 of file fasp_const.h.

9.27.2.117 #define SOLVER_SVGMRES 15

Variable-restart GMRES with safe net

Definition at line 131 of file fasp_const.h.

9.27.2.118 #define SOLVER_UMFPACK 32

UMFPack Direct Solver

Definition at line 139 of file fasp_const.h.

9.27.2.119 #define SOLVER_VFGMRES 6

Variable Restarting Flexible GMRES

Definition at line 123 of file fasp_const.h.

9.27.2.120 #define SOLVER_VGMRES 5

Variable Restarting GMRES

Definition at line 122 of file fasp_const.h.

9.27.2.121 #define STAG_RATIO 1e-4

Stagnation tolerance = tol*STAGRATIO

Definition at line 34 of file fasp_const.h.

9.27.2.122 #define STOP_MOD_REL_RES 3

modified relative residual $\|r\|/\|x\|$

Definition at line 147 of file fasp_const.h.

9.27.2.123 #define STOP_REL_PRECRES 2

relative B-residual $\|r\|_B / \|b\|_B$

Definition at line 146 of file fasp_const.h.

9.27.2.124 #define STOP_REL_RES 1

Definition of iterative solver stopping criteria types.

relative residual $\|r\| / \|b\|$

Definition at line 145 of file fasp_const.h.

9.27.2.125 #define TRUE 1

Definition of logic type.

logic TRUE

Definition at line 82 of file fasp_const.h.

9.27.2.126 #define UA_AMG 3

unsmoothed aggregation AMG

Definition at line 171 of file fasp_const.h.

9.27.2.127 #define UNPT -1

Undetermined points

Definition at line 226 of file fasp_const.h.

9.27.2.128 #define USERDEFINED 0

Type of ordering for smoothers.

User defined order

Definition at line 240 of file fasp_const.h.

9.27.2.129 #define V_CYCLE 1

Definition of cycle types.

V-cycle

Definition at line 182 of file fasp_const.h.

9.27.2.130 #define VMB 2

VMB aggregation

Definition at line 177 of file fasp_const.h.

9.27.2.131 #define W_CYCLE 2

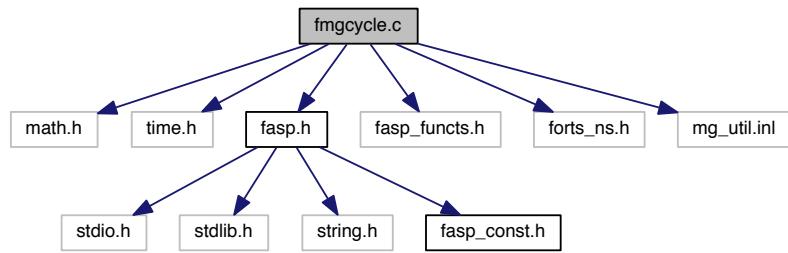
W-cycle

Definition at line 183 of file fasp_const.h.

9.28 fmfcycle.c File Reference

Abstract non-recursive full multigrid cycle.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "forts_ns.h"
#include "mg_util.inl"
Include dependency graph for fmfcycle.c:
```



Functions

- void [fasp_solver_fmfcycle](#) ([AMG_data](#) *mgl, [AMG_param](#) *param)
Solve Ax=b with non-recursive full multigrid K-cycle.

9.28.1 Detailed Description

Abstract non-recursive full multigrid cycle.

9.28.2 Function Documentation

9.28.2.1 void [fasp_solver_fmfcycle](#) ([AMG_data](#) * *mgl*, [AMG_param](#) * *param*)

Solve Ax=b with non-recursive full multigrid K-cycle.

Parameters

<i>mgl</i>	Pointer to AMG data: AMG_data
<i>param</i>	Pointer to AMG parameters: AMG_param

Author

Chensong Zhang

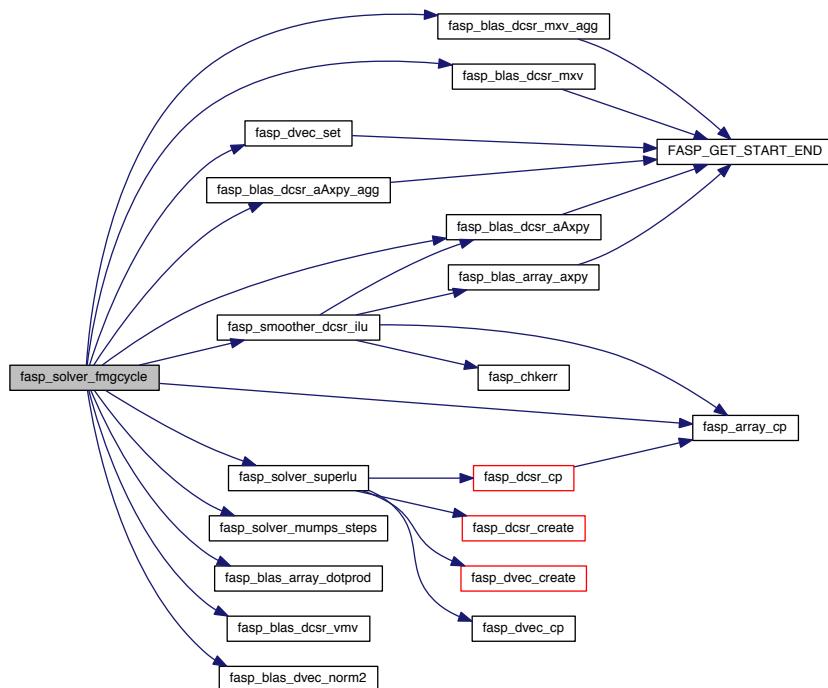
Date

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.

Definition at line 35 of file fmfcycle.c.

Here is the call graph for this function:

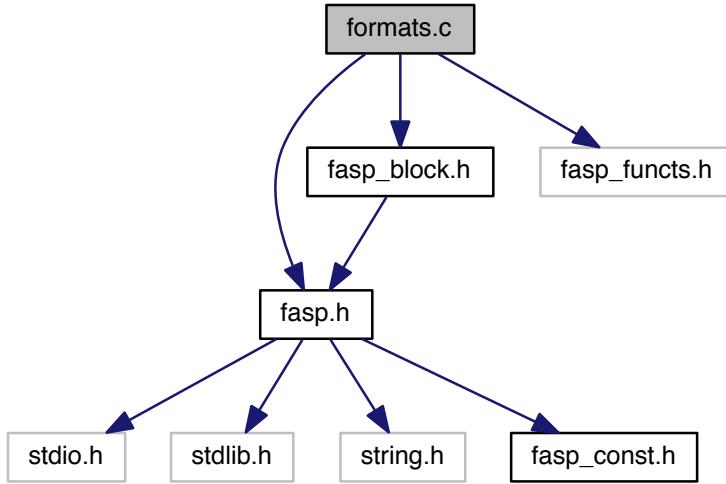


9.29 formats.c File Reference

Matrix format conversion routines.

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

Include dependency graph for formats.c:



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_bdcsr_dcsr (block_dCSRmat *Ab)**
Form the whole dCSRmat A using blocks given in Ab.
- **dCSRmat * 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, 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.

9.29.1 Detailed Description

Matrix format conversion routines.

9.29.2 Function Documentation

9.29.2.1 dCSRmat fasp_format_bdcsr_dcsr (block_dCSRmat * Ab)

Form the whole [dCSRmat](#) A using blocks given in Ab.

Parameters

<i>Ab</i>	Pointer to block_dCSRmat matrix
-----------	---

Returns

[dCSRmat](#) matrix if succeed, NULL if fail

Author

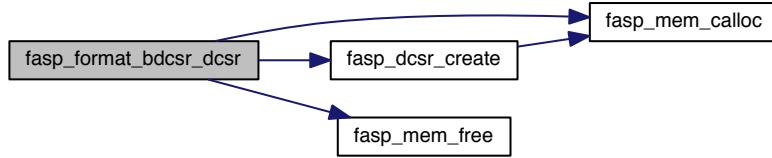
Shiquan Zhang

Date

08/10/2010

Definition at line 293 of file formats.c.

Here is the call graph for this function:



9.29.2.2 dCOOmat * fasp_format_dbsr_dcoo (dBsrmat * B)

Transfer a '[dBsrmat](#)' type matrix to a '[dCOOmat](#)' type matrix.

Parameters

<i>B</i>	Pointer to dBsrmat matrix
----------	---

Returns

Pointer to [dCOOmat](#) matrix

Author

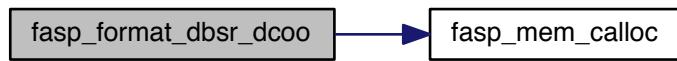
Zhiyang Zhou

Date

2010/10/26

Definition at line 944 of file formats.c.

Here is the call graph for this function:



9.29.2.3 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

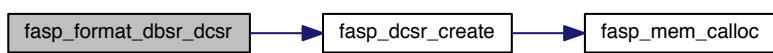
Modified by Chunsheng Feng, Xiaoqiang Yue on 05/24/2012

Note

Works for general nb (Xiaozhe)

Definition at line 496 of file formats.c.

Here is the call graph for this function:



9.29.2.4 SHORT fasp_format_dcoo_dcsr (dCOOMat * A, dCSRmat * B)

Transform a REAL matrix from its IJ format to its CSR format.

Parameters

<i>A</i>	Pointer to dCOOmat matrix
<i>B</i>	Pointer to dCSRmat matrix

Returns

FASP_SUCCESS if succeed

Author

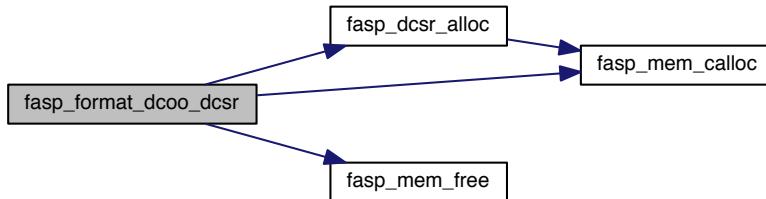
Xuehai Huang

Date

08/10/2009

Definition at line 28 of file formats.c.

Here is the call graph for this function:



9.29.2.5 [dBSRmat](#) `fasp_format_dcsr_dbsr(dCSRmat *A, INT nb)`

Transfer a [dCSRmat](#) type matrix into a [dBSRmat](#).

Parameters

<i>A</i>	Pointer to the dCSRmat type matrix
<i>nb</i>	size of each block

Returns

[dBSRmat](#) matrix

Author

Zheng Li

Date

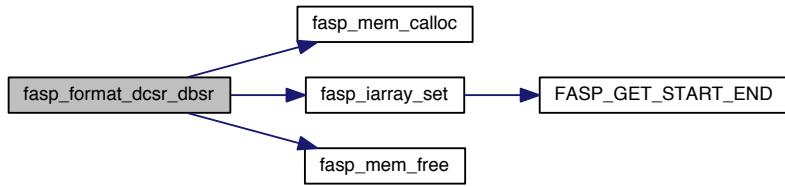
03/27/2014

Note

modified by Xiaozhe Hu to avoid potential memory leakage problem

Definition at line 722 of file formats.c.

Here is the call graph for this function:



9.29.2.6 **SHORT fasp_format_dcsr_dcoo(dCSRmat *A, dCOOmat *B)**

Transform a REAL matrix from its CSR format to its IJ format.

Parameters

<i>A</i>	Pointer to dCSRmat matrix
<i>B</i>	Pointer to dCOOmat matrix

Returns

FASP_SUCCESS if succeed

Author

Xuehai Huang

Date

08/10/2009

Modified by Chunsheng Feng, Zheng Li

Date

10/12/2012

Definition at line 81 of file formats.c.

Here is the call graph for this function:

**9.29.2.7 dCSRLmat * fasp_format_dcsrl_dcsr (dCSRmat * A)**Convert a [dCSRmat](#) into a [dCSRLmat](#).**Parameters**

<code>A</code>	Pointer to dCSRLmat matrix
----------------	--

ReturnsPointer to [dCSRLmat](#) matrix**Author**

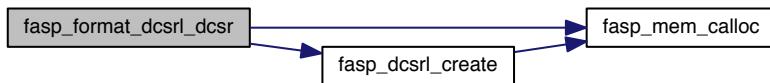
Zhiyang Zhou

Date

2011/01/07

Definition at line 362 of file formats.c.

Here is the call graph for this function:

**9.29.2.8 dBSRmat fasp_format_dstr_dbsr (dSTRmat * B)**Transfer a '[dSTRmat](#)' type matrix to a '[dBSRmat](#)' type matrix.

Parameters

<i>B</i>	Pointer to dSTRmat matrix
----------	----------------------------------

Returns**dBSRmat** matrix**Author**

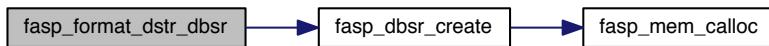
Zhiyang Zhou

Date

2010/10/26

Definition at line 840 of file formats.c.

Here is the call graph for this function:

**9.29.2.9 SHORT fasp_format_dstr_dcsr(**dSTRmat** * *A*, **dCSRmat** * *B*)**Transfer a '**dSTRmat**' type matrix into a '**dCSRmat**' type matrix.**Parameters**

<i>A</i>	Pointer to dSTRmat matrix
<i>B</i>	Pointer to dCSRmat matrix

Returns

FASP_SUCCESS if succeed

Author

Zhiyang Zhou

Date

2010/04/29

Definition at line 118 of file formats.c.

Here is the call graph for this function:

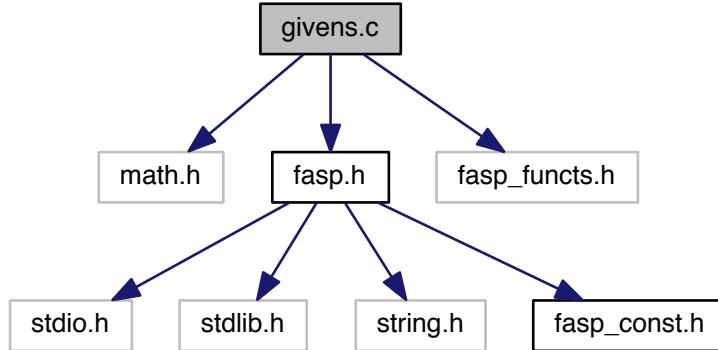


9.30 givens.c File Reference

Givens transformation.

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

Include dependency graph for givens.c:



Functions

- void `fasp_aux_givens` (const `REAL` beta, `dCSRmat` *H, `dvector` *y, `REAL` *tmp)
Perform Givens rotations to compute $y |beta \cdot e_1 - H \cdot y|$.

9.30.1 Detailed Description

Givens transformation.

9.30.2 Function Documentation

9.30.2.1 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

<i>beta</i>	Norm of residual r_0
<i>H</i>	Upper Hessenberg dCSRmat matrix: $(m+1) \times m$
<i>y</i>	Minimizer of $ beta*e_1 - H*y $
<i>tmp</i>	Temporary work array

Author

Xuehai Huang

Date

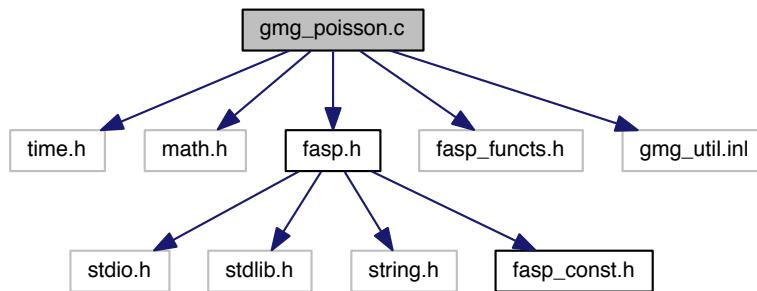
10/19/2008

Definition at line 28 of file givens.c.

9.31 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"
Include dependency graph for gmg_poisson.c:
```



Functions

- [INT fasp_poisson_gmg_1D \(REAL *u, REAL *b, INT nx, INT maxlevel, REAL rtol, const SHORT prtlvl\)](#)

Solve Ax=b of Poisson 1D equation by Geometric Multigrid Method.

- `INT fasp_poisson_gmg_2D (REAL *u, REAL *b, INT nx, INT ny, INT maxlevel, 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, INT nx, INT ny, INT nz, INT maxlevel, REAL rtol, const SHORT prtlvl)`

Solve Ax=b of Poisson 3D equation by Geometric Multigrid Method.

- `void fasp_poisson_fgm 1D (REAL *u, REAL *b, INT nx, INT maxlevel, REAL rtol, const SHORT prtlvl)`

Solve Ax=b of Poisson 1D equation by Geometric Multigrid Method (Full Multigrid)

- `void fasp_poisson_fgm 2D (REAL *u, REAL *b, INT nx, INT ny, INT maxlevel, REAL rtol, const SHORT prtlvl)`

Solve Ax=b of Poisson 2D equation by Geometric Multigrid Method (Full Multigrid)

- `void fasp_poisson_fgm 3D (REAL *u, REAL *b, INT nx, INT ny, INT nz, INT maxlevel, 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, INT nx, INT maxlevel, 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, INT nx, INT ny, INT maxlevel, 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, INT nx, INT ny, INT nz, INT maxlevel, REAL rtol, const SHORT prtlvl)`

Solve Ax=b of Poisson 3D equation by Geometric Multigrid Method (GMG preconditioned Conjugate Gradient method)

9.31.1 Detailed Description

GMG method as an iterative solver for Poisson Problem.

9.31.2 Function Documentation

9.31.2.1 void fasp_poisson_fgm 1D (`REAL * u, REAL * b, INT nx, INT maxlevel, REAL rtol, const SHORT prtlvl`)

Solve Ax=b of Poisson 1D equation by Geometric Multigrid Method (Full Multigrid)

Parameters

<code>u</code>	Pointer to the vector of dofs
<code>b</code>	Pointer to the vector of right hand side
<code>nx</code>	Number of grids in x direction
<code>maxlevel</code>	Maximum levels of the multigrid
<code>rtol</code>	Relative tolerance to judge convergence
<code>prtlvl</code>	Print level for output

Author

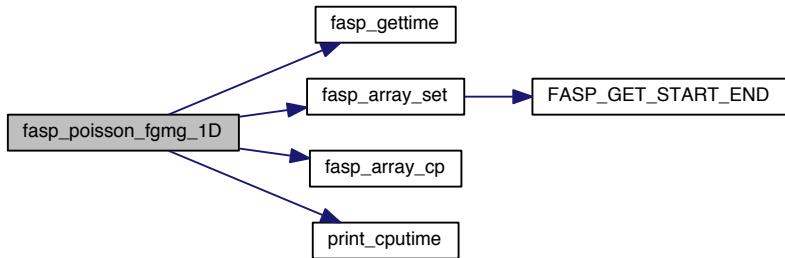
Ziteng Wang

Date

06/07/2013

Definition at line 417 of file gmg_poisson.c.

Here is the call graph for this function:



9.31.2.2 void fasp_poisson_fgm1D (REAL * u, REAL * b, INT nx, INT ny, INT maxlevel, REAL rtol, const SHORT prtlvl)

Solve Ax=b of Poisson 2D equation by Geometric Multigrid Method (Full Multigrid)

Parameters

<i>u</i>	Pointer to the vector of dofs
<i>b</i>	Pointer to the vector of right hand side
<i>nx</i>	Number of grids in x direction
<i>ny</i>	Number of grids in Y direction
<i>maxlevel</i>	Maximum levels of the multigrid
<i>rtol</i>	Relative tolerance to judge convergence
<i>prtlvl</i>	Print level for output

Author

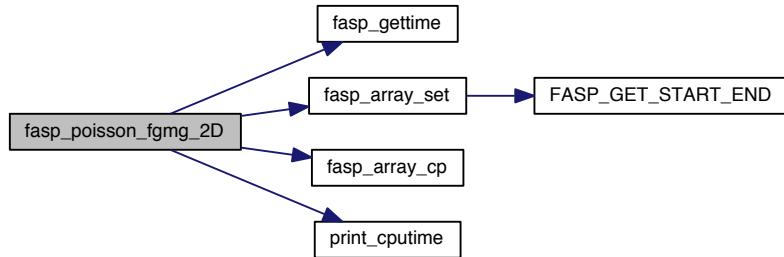
Ziteng Wang

Date

06/07/2013

Definition at line 510 of file gmg_poisson.c.

Here is the call graph for this function:



9.31.2.3 void fasp_poisson_fgmf_3D (REAL * *u*, REAL * *b*, INT *nx*, INT *ny*, INT *nz*, INT *maxlevel*, REAL *rtol*, const SHORT *prtlvl*)

Solve Ax=b of Poisson 3D equation by Geometric Multigrid Method (Full Multigrid)

Parameters

<i>u</i>	Pointer to the vector of dofs
<i>b</i>	Pointer to the vector of right hand side
<i>nx</i>	Number of grids in x direction
<i>ny</i>	NUmber of grids in y direction
<i>nz</i>	NUmber of grids in z direction
<i>maxlevel</i>	Maximum levels of the multigrid
<i>rtol</i>	Relative tolerance to judge convergence
<i>prtlvl</i>	Print level for output

Author

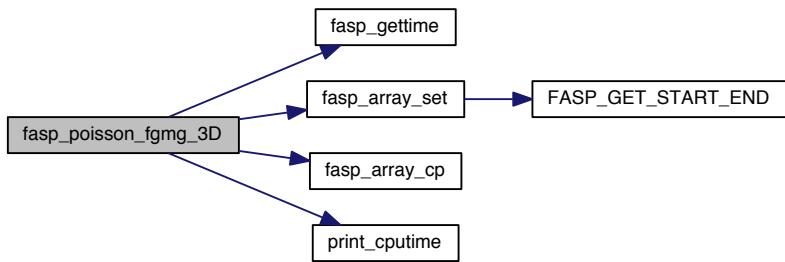
Ziteng Wang

Date

06/07/2013

Definition at line 617 of file gmg_poisson.c.

Here is the call graph for this function:

9.31.2.4 INT fasp_poisson_gmg_1D (REAL * *u*, REAL * *b*, INT *nx*, INT *maxlevel*, REAL *rtol*, const SHORT *prtlvl*)

Solve Ax=b of Poisson 1D equation by Geometric Multigrid Method.

Parameters

<i>u</i>	Pointer to the vector of dofs
<i>b</i>	Pointer to the vector of right hand side
<i>nx</i>	Number of grids in x direction
<i>maxlevel</i>	Maximum levels of the multigrid
<i>rtol</i>	Relative tolerance to judge convergence
<i>prtlvl</i>	Print level for output

Author

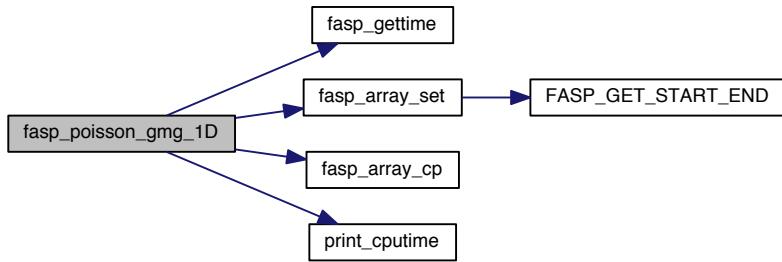
Ziteng Wang

Date

06/07/2013

Definition at line 34 of file gmg_poisson.c.

Here is the call graph for this function:

9.31.2.5 INT fasp_poisson_gmg_2D (REAL * *u*, REAL * *b*, INT *nx*, INT *ny*, INT *maxlevel*, REAL *rtol*, const SHORT *prtlvl*)

Solve Ax=b of Poisson 2D equation by Geometric Multigrid Method.

Parameters

<i>u</i>	Pointer to the vector of dofs
<i>b</i>	Pointer to the vector of right hand side
<i>nx</i>	Number of grids in x direction
<i>ny</i>	Number of grids in y direction
<i>maxlevel</i>	Maximum levels of the multigrid
<i>rtol</i>	Relative tolerance to judge convergence
<i>prtlvl</i>	Print level for output

Author

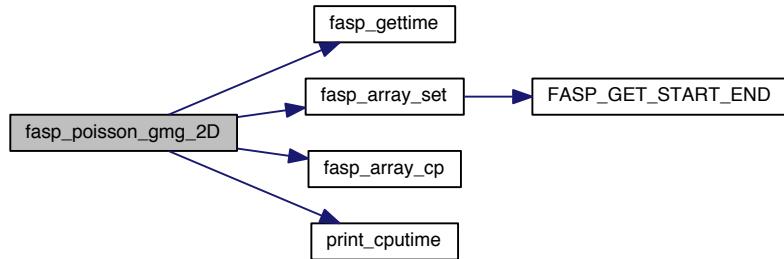
Ziteng Wang

Date

06/07/2013

Definition at line 155 of file gmg_poisson.c.

Here is the call graph for this function:



9.31.2.6 INT fasp_poisson_gmg_3D (REAL * *u*, REAL * *b*, INT *nx*, INT *ny*, INT *nz*, INT *maxlevel*, REAL *rtol*, const SHORT *prtlvl*)

Solve Ax=b of Poisson 3D equation by Geometric Multigrid Method.

Parameters

<i>u</i>	Pointer to the vector of dofs
<i>b</i>	Pointer to the vector of right hand side
<i>nx</i>	Number of grids in x direction
<i>ny</i>	Number of grids in y direction
<i>nz</i>	Number of grids in z direction
<i>maxlevel</i>	Maximum levels of the multigrid
<i>rtol</i>	Relative tolerance to judge convergence
<i>prtlvl</i>	Print level for output

Author

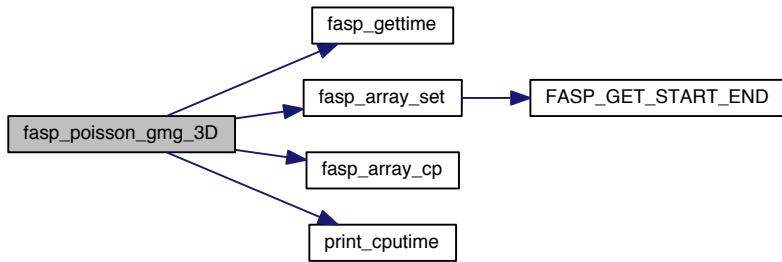
Ziteng Wang

Date

06/07/2013

Definition at line 286 of file gmg_poisson.c.

Here is the call graph for this function:



9.31.2.7 INT fasp_poisson_pcg_gmg_1D (REAL * *u*, REAL * *b*, INT *nx*, INT *maxlevel*, REAL *rtol*, const SHORT *prtlvl*)

Solve Ax=b of Poisson 1D equation by Geometric Multigrid Method (GMG preconditioned Conjugate Gradient method)

Parameters

<i>u</i>	Pointer to the vector of dofs
<i>b</i>	Pointer to the vector of right hand side
<i>nx</i>	Number of grids in x direction
<i>maxlevel</i>	Maximum levels of the multigrid
<i>rtol</i>	Relative tolerance to judge convergence
<i>prtlvl</i>	Print level for output

Author

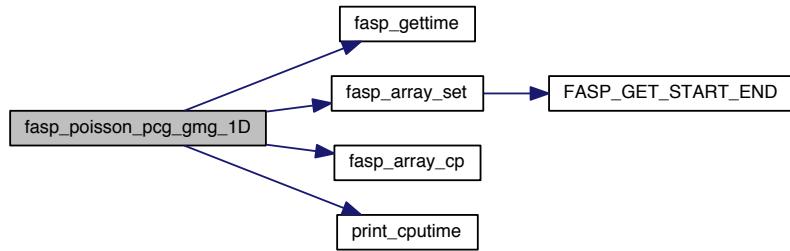
Ziteng Wang

Date

06/07/2013

Definition at line 724 of file gmg_poisson.c.

Here is the call graph for this function:



9.31.2.8 INT fasp_poisson_pcg_gmg_2D (REAL * *u*, REAL * *b*, INT *nx*, INT *ny*, INT *maxlevel*, REAL *rtol*, const SHORT *prtlvl*)

Solve Ax=b of Poisson 2D equation by Geometric Multigrid Method (GMG preconditioned Conjugate Gradient method)

Parameters

<i>u</i>	Pointer to the vector of dofs
<i>b</i>	Pointer to the vector of right hand side
<i>nx</i>	Number of grids in x direction
<i>ny</i>	Number of grids in y direction
<i>maxlevel</i>	Maximum levels of the multigrid
<i>rtol</i>	Relative tolerance to judge convergence
<i>prtlvl</i>	Print level for output

Author

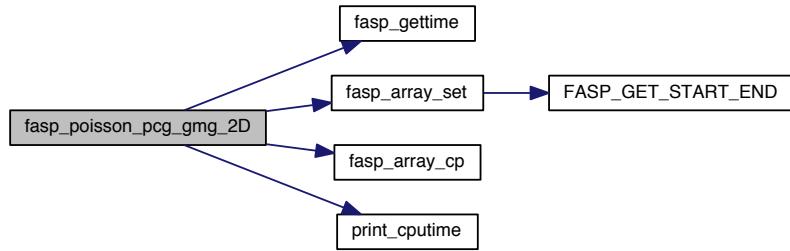
Ziteng Wang

Date

06/07/2013

Definition at line 815 of file gmg_poisson.c.

Here is the call graph for this function:



9.31.2.9 INT fasp_poisson_pcg_gmg_3D (REAL * *u*, REAL * *b*, INT *nx*, INT *ny*, INT *nz*, INT *maxlevel*, REAL *rtol*, const SHORT *prtlvl*)

Solve Ax=b of Poisson 3D equation by Geometric Multigrid Method (GMG preconditioned Conjugate Gradient method)

Parameters

<i>u</i>	Pointer to the vector of dofs
<i>b</i>	Pointer to the vector of right hand side
<i>nx</i>	Number of grids in x direction
<i>ny</i>	Number of grids in y direction
<i>nz</i>	Number of grids in z direction
<i>maxlevel</i>	Maximum levels of the multigrid
<i>rtol</i>	Relative tolerance to judge convergence
<i>prtlvl</i>	Print level for output

Author

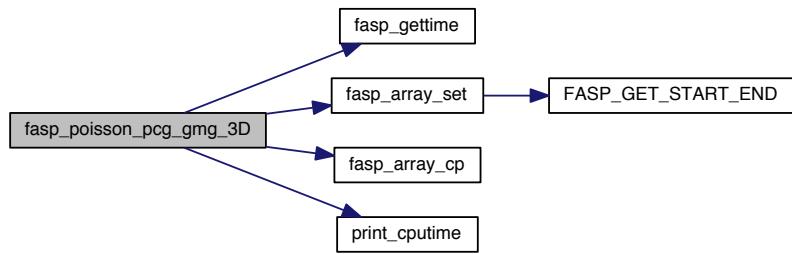
Ziteng Wang

Date

06/07/2013

Definition at line 921 of file gmg_poisson.c.

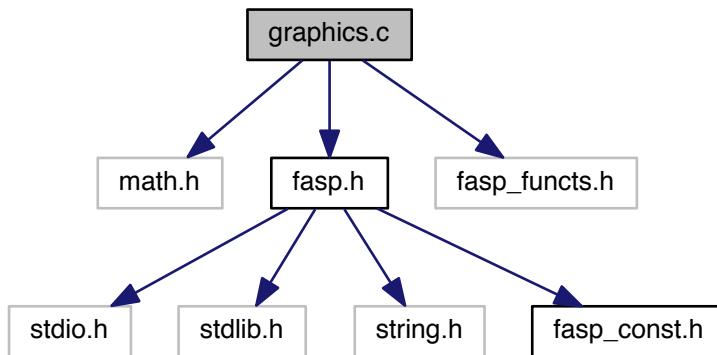
Here is the call graph for this function:



9.32 graphics.c File Reference

Functions for graphical output.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
Include dependency graph for graphics.c:
```



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.

9.32.1 Detailed Description

Functions for graphical output.

9.32.2 Function Documentation

9.32.2.1 void `fasp_dbsr_plot` (const `dBSRmat` * A, const char * *filename*)

Write dBSR sparse matrix pattern in BMP file format.

Parameters

A	Pointer to the <code>dBSRmat</code> matrix
<i>filename</i>	File name

Author

Chunsheng Feng

Date

11/16/2013

Note

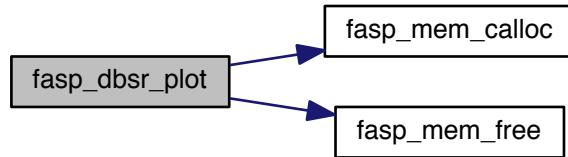
The routine `fasp_dbsr_plot` writes pattern of the specified `dBSRmat` matrix in uncompressed BMP file format (Windows bitmap) to a binary file whose name is specified by the character string *filename*.

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 462 of file graphics.c.

Here is the call graph for this function:



9.32.2.2 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 <code>dBSRmat</code> matrix
<i>filename</i>	File name
<i>size</i>	<i>size*size</i> is the picture size for the picture

Author

Chunsheng Feng

Date

11/16/2013

Note

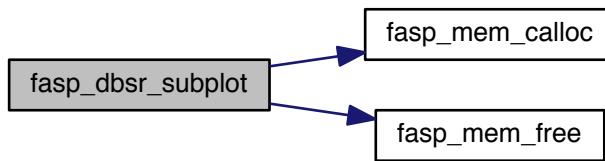
The routine `fasp_dbsr_subplot` writes pattern of the specified `dBSRmat` matrix in uncompressed BMP file format (Windows bitmap) to a binary file whose name is specified by the character string filename.

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`.

Here is the call graph for this function:



9.32.2.3 INT fasp_dcsr_plot (const dCSRmat * A, const char * fname)

Write dCSR sparse matrix pattern in BMP file format.

Parameters

<code>A</code>	Pointer to the <code>dBSRmat</code> matrix
<code>fname</code>	File name to plot to

Author

Chunsheng Feng

Date

11/16/2013

Note

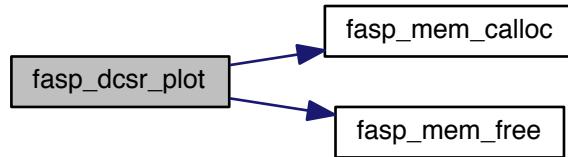
The routine `fasp_dcsr_plot` writes pattern of the specified `dCSRmat` matrix in uncompressed BMP file format (Windows bitmap) to a binary file whose name is specified by the character string filename.

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 622 of file `graphics.c`.

Here is the call graph for this function:



9.32.2.4 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
<i>filename</i>	File name
<i>size</i>	<i>size*size</i> is the picture size for the picture

Author

Chensong Zhang

Date

03/29/2009

Note

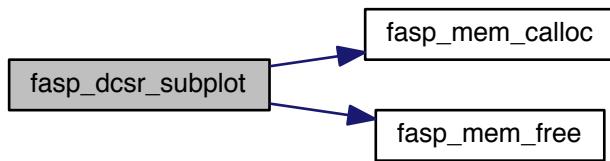
The routine `fasp_dcsr_subplot` writes pattern of the specified `dCSRmat` matrix in uncompressed BMP file format (Windows bitmap) to a binary file whose name is specified by the character string filename.

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`.

Here is the call graph for this function:



9.32.2.5 void fasp_grid2d_plot(pgrid2d pg, INT level)

Output grid to a EPS file.

Parameters

<i>pg</i>	Pointer to grid in 2d
<i>level</i>	Number of levels

Author

Chensong Zhang

Date

03/29/2009

Definition at line 172 of file `graphics.c`.

9.33 ilu.f File Reference

ILU routines for preconditioning adapted from SPARSEKIT.

Functions/Subroutines

- subroutine `iluk` (*n*, *a*, *ja*, *ia*, *lfil*, *alu*, *jlu*, *iwk*, *ierr*, *nzlu*)

- subroutine **ilut** (n, a, ja, ia, lfil, droptol, alu, jlu, iwk, ierr, nz)
- subroutine **ilutp** (n, a, ja, ia, lfil, droptol, permtol, mbloc, alu, jlu, iwk, ierr, nz)
- subroutine **sstr** (num, q)
- subroutine **qsplit** (a, ind, n, ncut)
- subroutine **symbfactor** (n, colind, rwptra, levfill, nzmax, nzlu, ijlu, uptr, ierr)

9.33.1 Detailed Description

ILU routines for preconditioning adapted from SPARSEKIT.

Note

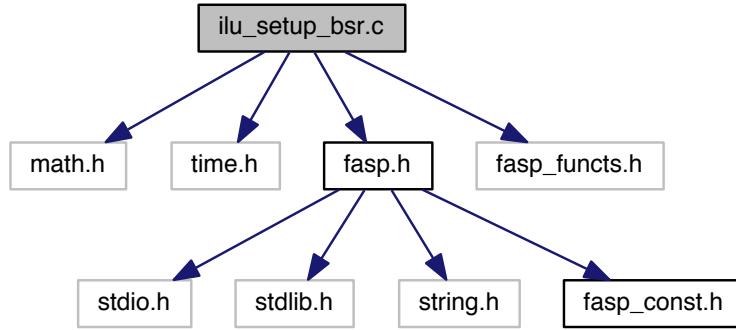
Incomplete Factorization Methods: ILUk, ILUt, ILUtp

9.34 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"
```

Include dependency graph for `ilu_setup_bsr.c`:



Functions

- void **symbfactor_** (const [INT](#) *n, [INT](#) *colind, [INT](#) *rwptra, 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 decomposition of a BSR matrix A.

9.34.1 Detailed Description

Setup Incomplete LU decomposition for [dBSRmat](#) matrices.

9.34.2 Function Documentation

9.34.2.1 SHORT fasp_ilu_dbsr_setup ([dBSRmat](#) * A, [ILU_data](#) * *iludata*, [ILU_param](#) * *iluparam*)

Get ILU decomposition of a BSR matrix A.

Parameters

<i>A</i>	Pointer to dBSRmat matrix
<i>iludata</i>	Pointer to ILU_data
<i>iluparam</i>	Pointer to ILU_param

Author

Shiquan Zhang, Xiaozhe Hu

Date

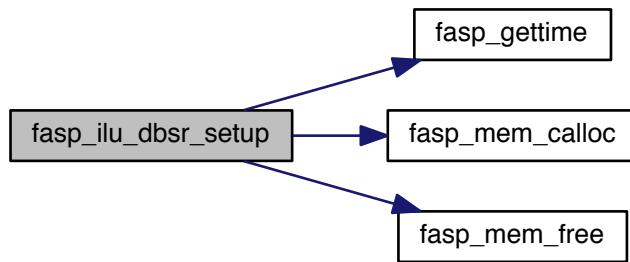
11/08/2010

Note

Works for general nb (Xiaozhe)

Definition at line 42 of file [ilu_setup_bsr.c](#).

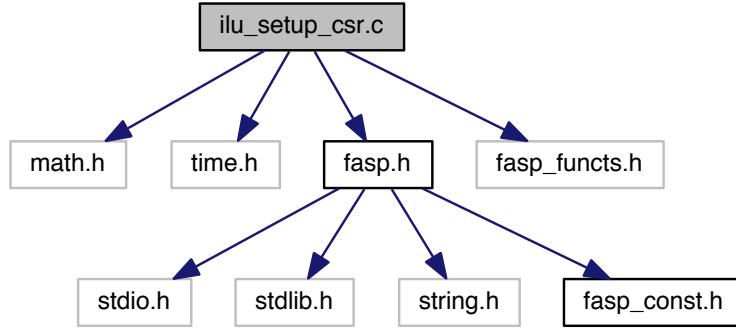
Here is the call graph for this function:



9.35 ilu_setup_csr.c File Reference

Setup of ILU decomposition for [dCSRmat](#) matrices.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
Include dependency graph for ilu_setup_csr.c:
```



Functions

- void **iluk_** (const INT *n, REAL *a, INT *ja, INT *ia, INT *lfil, REAL *alu, INT *jlu, INT *iwrk, 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 *iwrk, 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 *iwrk, 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.

9.35.1 Detailed Description

Setup of ILU decomposition for **dCSRmat** matrices.

9.35.2 Function Documentation

9.35.2.1 **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
---	----------------------------------

<i>iludata</i>	Pointer to ILU_data
<i>iluparam</i>	Pointer to ILU_param

Author

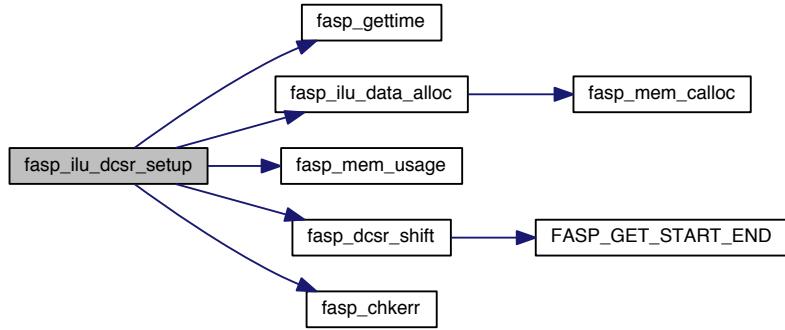
Shiquan Zhang Xiaozhe Hu

Date

12/27/2009

Definition at line 48 of file [ilu_setup_csr.c](#).

Here is the call graph for this function:

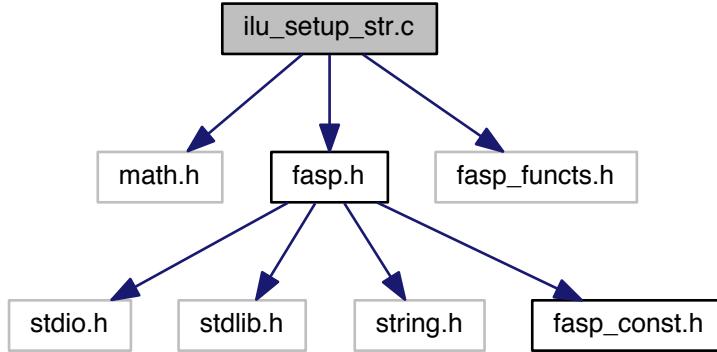


9.36 ilu_setup_str.c File Reference

Setup of ILU decomposition for [dSTRmat](#) matrices.

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

Include dependency graph for ilu_setup_str.c:



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) decomposition of a structured matrix A.

9.36.1 Detailed Description

Setup of ILU decomposition for `dSTRmat` matrices.

9.36.2 Function Documentation

9.36.2.1 void [fasp_ilu_dstr_setup0](#) (`dSTRmat * A, dSTRmat * LU`)

Get ILU(0) decomposition of a structured matrix A.

Parameters

<code>A</code>	Pointer to <code>dSTRmat</code>
<code>LU</code>	Pointer to ILU structured matrix of REAL type

Author

Shiquan Zhang, Xiaozhe Hu

Date

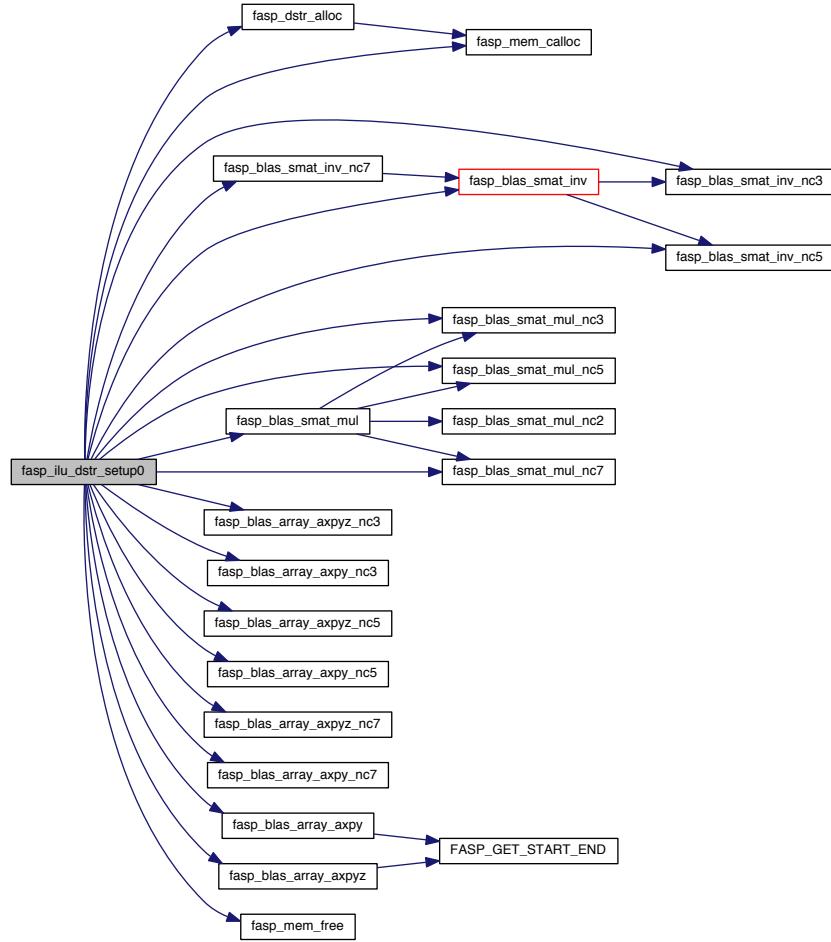
11/08/2010

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.

Here is the call graph for this function:



9.36.2.2 void fasp_ilu_dstr_setup1 (dSTRmat * A, dSTRmat * LU)

Get ILU(1) decomposition of a structured matrix A.

Parameters

A	Pointer to original structured matrix of REAL type
---	--

<i>LU</i>	Pointer to ILU structured matrix of REAL type
-----------	---

Author

Shiquan Zhang, Xiaozhe Hu

Date

11/08/2010

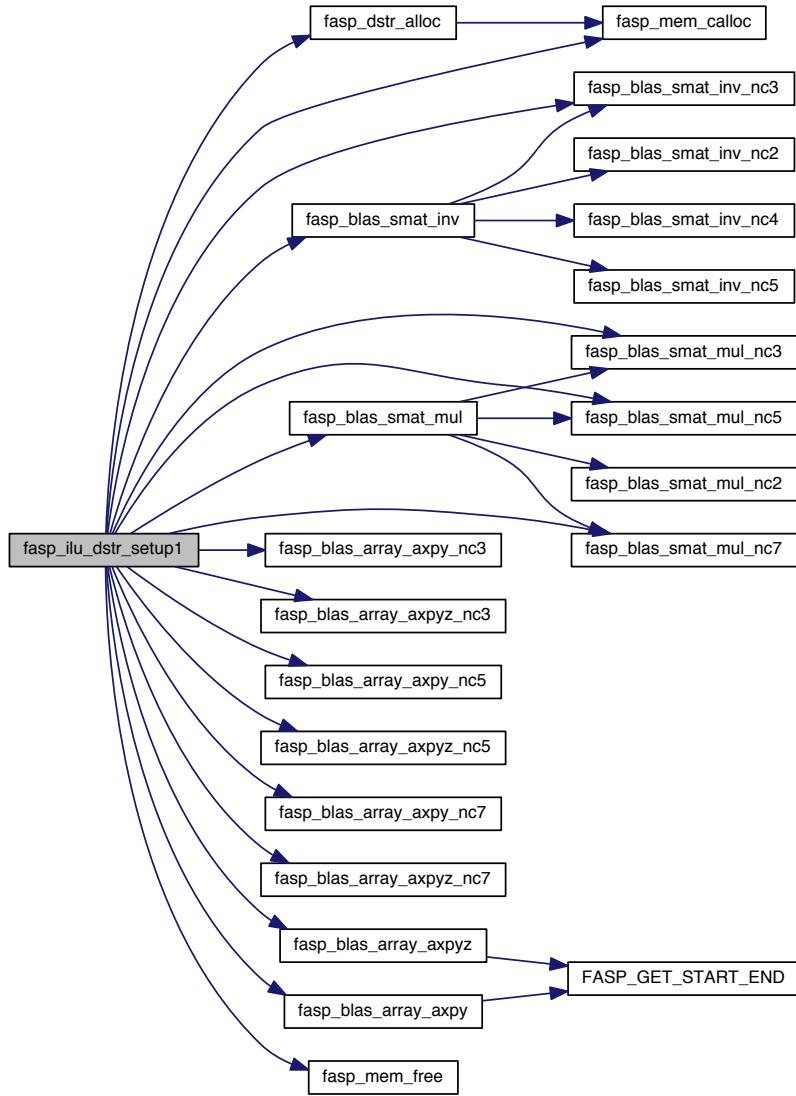
Note

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 319 of file `ilu_setup_str.c`.

Here is the call graph for this function:

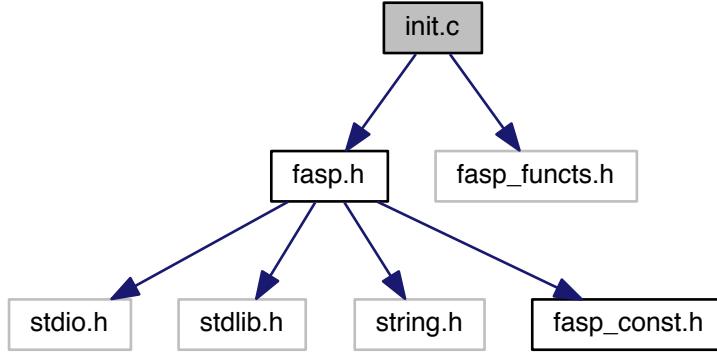


9.37 init.c File Reference

Initialize important data structures.

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

Include dependency graph for init.c:



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` (`INT` iwk, `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.

9.37.1 Detailed Description

Initialize important data structures.

Note

Every structures should be initialized before usage.

9.37.2 Function Documentation

9.37.2.1 `AMG_data_bsr * fasp_amg_data_bsr_create (SHORT max_levels)`

Create and initialize [AMG_data](#) data structure for AMG/SAMG (BSR format)

Parameters

<i>max_levels</i>	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.

Here is the call graph for this function:



9.37.2.2 `void fasp_amg_data_bsr_free (AMG_data_bsr * mgl)`

Free [AMG_data_bsr](#) data memory space.

Parameters

<i>mgl</i>	Pointer to the AMG_data_bsr
------------	---

Author

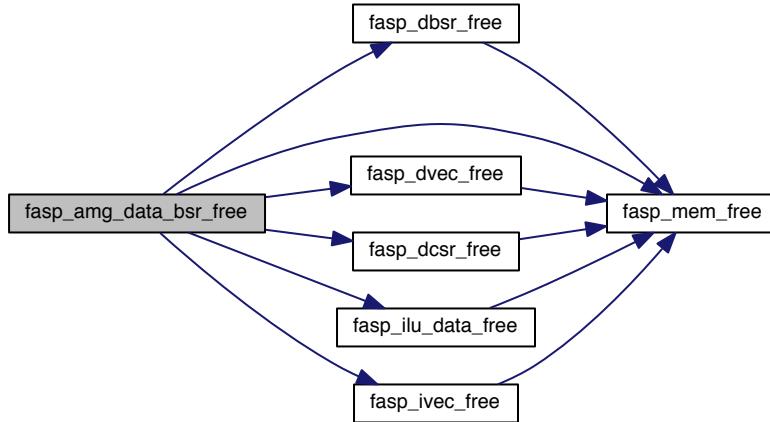
Xiaozhe Hu

Date

2013/02/13

Definition at line 241 of file init.c.

Here is the call graph for this function:

**9.37.2.3 AMG_data * fasp_amg_data_create (**SHORT** max_levels)**Create and initialize [AMG_data](#) for classical and SA AMG.**Parameters**

<i>max_levels</i>	Max number of levels allowed
-------------------	------------------------------

ReturnsPointer to the [AMG_data](#) data structure**Author**

Chensong Zhang

Date

2010/04/06

Definition at line 56 of file init.c.

Here is the call graph for this function:



9.37.2.4 void fasp_amg_data_free (AMG_data * mgl, AMG_param * param)

Free [AMG_data](#) data memory space.

Parameters

<i>mgl</i>	Pointer to the AMG_data
<i>param</i>	Pointer to AMG parameters

Author

Chensong Zhang

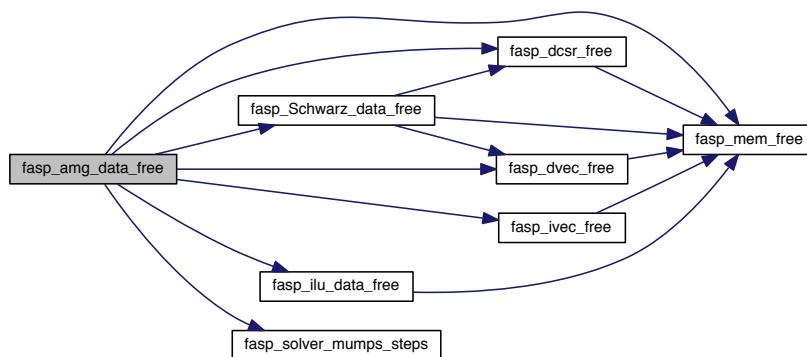
Date

2010/04/06

Modified by Chensong Zhang on 05/05/2013: Clean up param as well!

Definition at line 183 of file init.c.

Here is the call graph for this function:



9.37.2.5 void fasp_ilu_data_alloc (INT *iwk*, INT *nwork*, ILU_data * *iludata*)

Allocate workspace for ILU factorization.

Parameters

<i>iwk</i>	Size of the index array
<i>nwork</i>	Size of the work array
<i>iludata</i>	Pointer to the ILU_data

Author

Chensong Zhang

Date

2010/04/06

Definition at line 117 of file init.c.

Here is the call graph for this function:



9.37.2.6 void fasp_ilu_data_free (ILU_data * *ILUdata*)

Create [ILU_data](#) sturcture.

Parameters

<i>ILUdata</i>	Pointer to ILU_data
----------------	-------------------------------------

Author

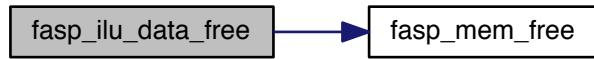
Chensong Zhang

Date

2010/04/03

Definition at line 286 of file init.c.

Here is the call graph for this function:

**9.37.2.7 void fasp_ilu_data_null (*ILU_data* * *ILUdata*)**

Initialize ILU data.

Parameters

<i>ILUdata</i>	Pointer to ILU_data
----------------	-------------------------------------

Author

Chensong Zhang

Date

2010/03/23

Definition at line 307 of file init.c.

9.37.2.8 void fasp_precond_data_null (*precond_data* * *pcdata*)

Initialize [precond_data](#).

Parameters

<i>pcdata</i>	Preconditioning data structure
---------------	--------------------------------

Author

Chensong Zhang

Date

2010/03/23

Definition at line 25 of file init.c.

9.37.2.9 void fasp_precond_null (*precond* * *pcdata*)

Initialize precond data.

Parameters

<code>pcdata</code>	Pointer to precond
---------------------	--------------------

Author

Chensong Zhang

Date

2010/03/23

Definition at line 323 of file init.c.

9.37.2.10 void fasp_Schwarz_data_free (Schwarz_data * Schwarz)

Free `Schwarz_data` data memory space.

Parameters

<code>*Schwarz</code>	pointer to the <code>AMG_data</code> data
-----------------------	---

Author

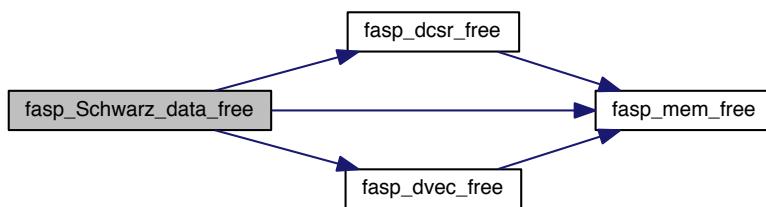
Xiaozhe Hu

Date

2010/04/06

Definition at line 143 of file init.c.

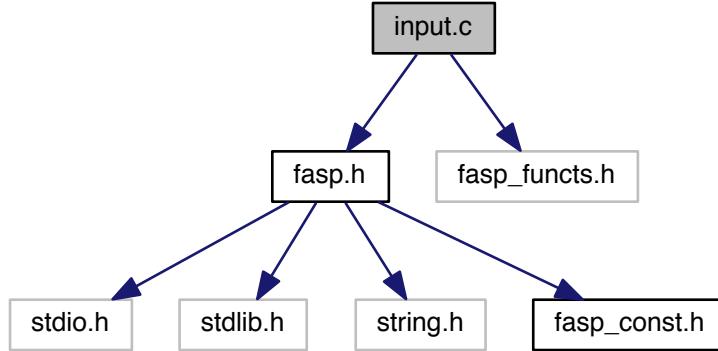
Here is the call graph for this function:



9.38 input.c File Reference

Read input parameters.

```
#include "fasp.h"
#include "fasp_functs.h"
Include dependency graph for input.c:
```



Functions

- **[SHORT fasp_param_check \(input_param *inparam\)](#)**
Simple check on input parameters.
- **[void fasp_param_input \(const char *filenm, input_param *inparam\)](#)**
Read input parameters from disk file.

9.38.1 Detailed Description

Read input parameters.

9.38.2 Function Documentation

9.38.2.1 **[SHORT fasp_param_check \(input_param * inparam \)](#)**

Simple check on input parameters.

Parameters

<i>inparam</i>	Input parameters
----------------	------------------

Author

Chensong Zhang

Date

09/29/2013

Definition at line 23 of file input.c.

9.38.2.2 void fasp_param_input (const char * *filenm*, input_param * *inparam*)

Read input parameters from disk file.

Parameters

<i>filenm</i>	File name for input file
<i>inparam</i>	Input parameters

Author

Chensong Zhang

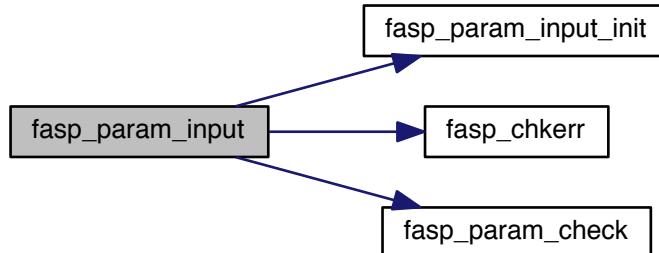
Date

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.

Definition at line 99 of file input.c.

Here is the call graph for this function:

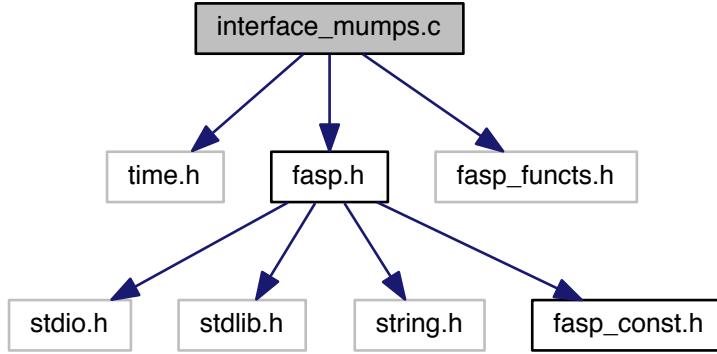


9.39 interface_mumps.c File Reference

Interface to MUMPS direct solvers.

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

Include dependency graph for interface_mumps.c:



Functions

- int [fasp_solver_mumps](#) (`dCSRmat *ptrA, dvector *b, dvector *u, const int print_level)`
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.

9.39.1 Detailed Description

Interface to MUMPS direct solvers.

9.39.2 Function Documentation

9.39.2.1 int [fasp_solver_mumps](#) (`dCSRmat * ptrA, dvector * b, dvector * u, const int print_level)`

Solve Ax=b by MUMPS directly.

Parameters

<code>ptrA</code>	Pointer to a <code>dCSRmat</code> matrix
<code>b</code>	Pointer to the dvector of right-hand side term
<code>u</code>	Pointer to the dvector of solution
<code>print_level</code>	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 35 of file interface_mumps.c.

9.39.2.2 int fasp_solver_mumps_steps (dCSRmat * *ptrA*, dvector * *b*, dvector * *u*, Mumps_data * *mumps*)

Solve Ax=b by MUMPS in three steps.

Parameters

<i>ptrA</i>	Pointer to a dCSRmat matrix
<i>b</i>	Pointer to the dvector of right-hand side term
<i>u</i>	Pointer to the dvector of solution
<i>mumps</i>	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.

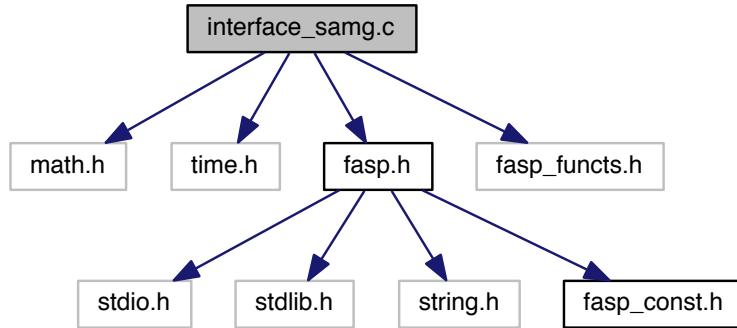
Definition at line 163 of file interface_mumps.c.

9.40 interface_samg.c File Reference

Interface to SAMG.

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

Include dependency graph for interface_samg.c:



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.

9.40.1 Detailed Description

Interface to SAMG.

Add reference for SAMG by K. Stuben here!

9.40.2 Function Documentation

9.40.2.1 INT [dCSRmat2SAMGInput](#) (dCSRmat * A, char * filefrm, char * fileamg)

Write SAMG Input data from a sparse matrix of CSR format.

Parameters

<code>*A</code>	pointer to the dCSRmat matrix
<code>*filefrm</code>	pointer to the name of the .frm file
<code>*fileamg</code>	pointer to the name of the .amg file

Author

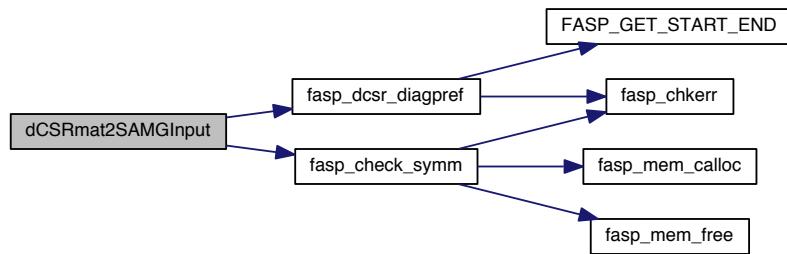
Zhiyang Zhou

Date

2010/08/25

Definition at line 56 of file interface_samg.c.

Here is the call graph for this function:



9.40.2.2 void dvector2SAMGInput (`dvector` * `vec`, `char` * `filename`)

Write a dvector to disk file in SAMG format (coordinate format)

Parameters

<code>*vec</code>	pointer to the dvector
<code>*filename</code>	char for vector file name

Author

Zhiyang Zhou

Date

08/25/2010

Definition at line 27 of file interface_samg.c.

Here is the call graph for this function:

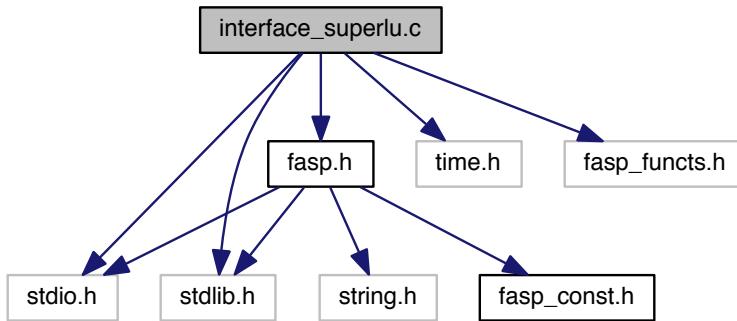


9.41 interface_superlu.c File Reference

Interface to SuperLU direct solvers.

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

Include dependency graph for interface_superlu.c:



Functions

- int `fasp_solver_superlu (dCSRmat *ptrA, dvector *b, dvector *u, const int print_level)`
Solve $Au=b$ by SuperLU.

9.41.1 Detailed Description

Interface to SuperLU direct solvers.

9.41.2 Function Documentation

9.41.2.1 int `fasp_solver_superlu (dCSRmat * ptrA, dvector * b, dvector * u, const int print_level)`

Solve $Au=b$ by SuperLU.

Parameters

<code>ptrA</code>	Pointer to a <code>dCSRmat</code> matrix
<code>b</code>	Pointer to the <code>dvector</code> of right-hand side term

<i>u</i>	Pointer to the dvector of solution
<i>print_level</i>	Output level

Author

Xiaozhe Hu

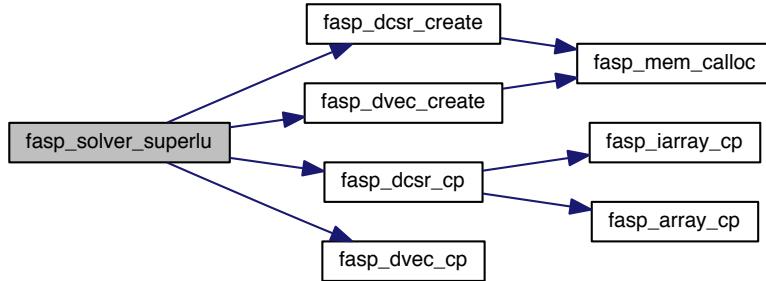
Date

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 39 of file interface_superlu.c.

Here is the call graph for this function:

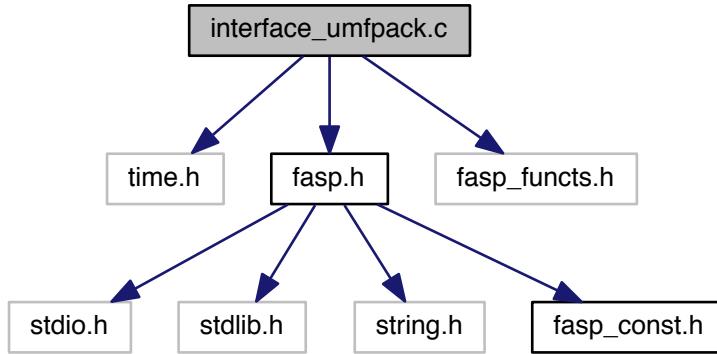


9.42 interface_umfpack.c File Reference

Interface to UMFPACK direct solvers.

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

Include dependency graph for interface_umfpack.c:



Functions

- `INT fasp_solver_umfpack (dCSRmat *ptrA, dvector *b, dvector *u, const INT print_level)`
Solve $Au=b$ by UMFPack.

9.42.1 Detailed Description

Interface to UMFPACK direct solvers.

9.42.2 Function Documentation

9.42.2.1 INT fasp_solver_umfpack (`dCSRmat * ptrA, dvector * b, dvector * u, const INT print_level)`

Solve $Au=b$ by UMFPack.

Parameters

<code>ptrA</code>	Pointer to a <code>dCSRmat</code> matrix
<code>b</code>	Pointer to the dvector of right-hand side term
<code>u</code>	Pointer to the dvector of solution
<code>print_level</code>	Output level

Author

Chensong Zhang

Date

05/20/2010

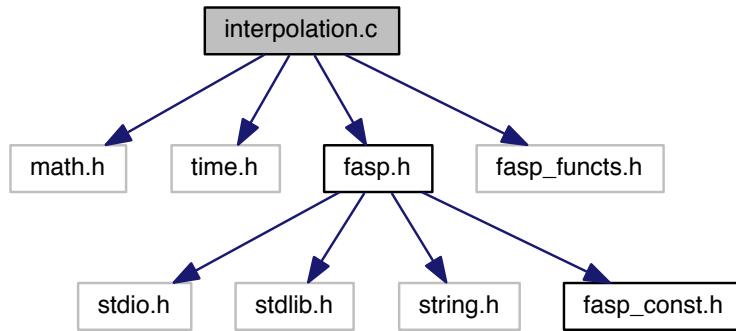
Modified by Chensong Zhang on 02/27/2013 for new FASP function names.

Definition at line 34 of file interface_umfpack.c.

9.43 interpolation.c File Reference

Interpolation operators for AMG.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
Include dependency graph for interpolation.c:
```



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.

9.43.1 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.

9.43.2 Function Documentation

9.43.2.1 void fasp_amg_interp (**dCSRmat** * *A*, **ivector** * *vertices*, **dCSRmat** * *P*, **iCSRmat** * *S*, **AMG_param** * *param*)

Generate interpolation operator P.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix (index starts from 0)
<i>vertices</i>	Indicator vector for the C/F splitting of the variables
<i>P</i>	Prolongation (input: nonzero pattern, output: prolongation)
<i>S</i>	Strong connection matrix
<i>param</i>	AMG parameters

Author

Xuehai Huang, Chensong Zhang

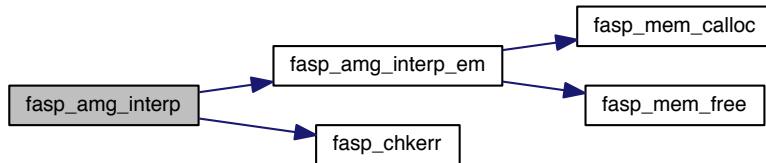
Date

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.

Here is the call graph for this function:



9.43.2.2 void fasp_amg_interp1 (**dCSRmat** * *A*, **ivector** * *vertices*, **dCSRmat** * *P*, **AMG_param** * *param*, **iCSRmat** * *S*, **INT** * *icor_ysk*)

Generate interpolation operator P.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix (index starts from 0)
<i>vertices</i>	Indicator vector for the C/F splitting of the variables
<i>P</i>	Prolongation (input: nonzero pattern, output: prolongation)
<i>S</i>	Strong connection matrix
<i>param</i>	AMG parameters
<i>icor_ysk</i>	Indices of coarse nodes in fine grid

Returns

FASP_SUCCESS or error message

Author

Chunsheng Feng, Xiaoqiang Yue

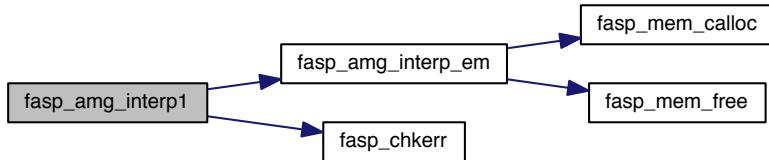
Date

03/01/2011

Modified by Chensong Zhang on 05/14/2013: reconstruct the code

Definition at line 105 of file interpolation.c.

Here is the call graph for this function:



9.43.2.3 void fasp_amg_interp_trunc (*dCSRmat* * *P*, *AMG_param* * *param*)

Truncation step for prolongation operators.

Parameters

<i>P</i>	Prolongation (input: full, output: truncated)
<i>param</i>	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.

Here is the call graph for this function:

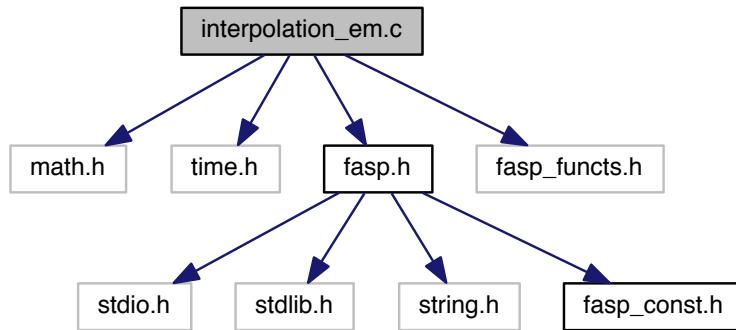


9.44 interpolation_em.c File Reference

Interpolation operators for AMG based on energy-min.

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

Include dependency graph for interpolation_em.c:



Functions

- void `fasp_amg_interp_em (dCSRmat *A, ivec *vertices, dCSRmat *P, AMG_param *param)`
Energy-min interpolation.

9.44.1 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

9.44.2 Function Documentation

9.44.2.1 void fasp_amg_interp_em (*dCSRmat* * *A*, *ivector* * *vertices*, *dCSRmat* * *P*, *AMG_param* * *param*)

Energy-min interpolation.

Parameters

<i>A</i>	Pointer to <i>dCSRmat</i> : the coefficient matrix (index starts from 0)
<i>vertices</i>	Pointer to the indicator of CF splitting on fine or coarse grid
<i>P</i>	Pointer to the <i>dCSRmat</i> matrix of resulted interpolation
<i>param</i>	Pointer to <i>AMG_param</i> : AMG parameters

Author

Shuo Zhang, Xuehai Huang

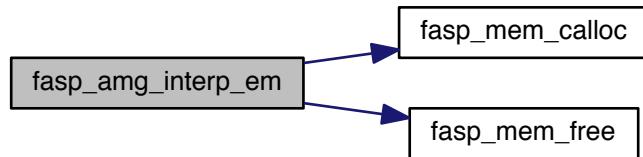
Date

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.

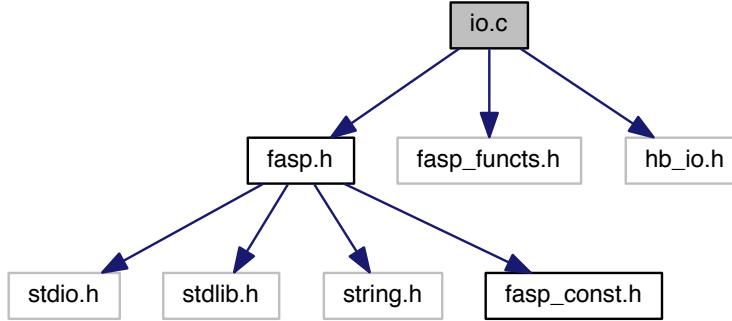
Here is the call graph for this function:



9.45 io.c File Reference

Matrix-vector input/output subroutines.

```
#include "fasp.h"
#include "fasp_functs.h"
#include "hb_io.h"
Include dependency graph for io.c:
```



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 0.
- 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` (char *input_file, `dCSRmat` *A, `dvector` *b)
Read matrix and right-hans side from a HB format file.

Variables

- INT ilength
- INT dlengt

9.45.1 Detailed Description

Matrix-vector input/output subroutines.

Note

Read, write or print a matrix or a vector in various formats.

9.45.2 Function Documentation

9.45.2.1 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

Date

11/16/2013

Definition at line 1445 of file io.c.

9.45.2.2 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

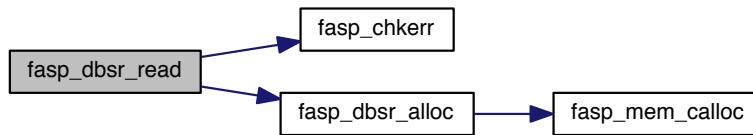
Xiaozhe Hu

Date

10/29/2010

Definition at line 691 of file io.c.

Here is the call graph for this function:



9.45.2.3 void fasp_dbsr_write (const char * *filename*, [dBSRmat](#) * *A*)

Write a [dBSRmat](#) to a disk file.

Parameters

<i>filename</i>	File name for <i>A</i>
<i>A</i>	Pointer to the dBSRmat matrix <i>A</i>

Note

The routine writes the specified REAL vector in BSR format.
Refer to the reading subroutine \ref fasp_dbsr_read.

Author

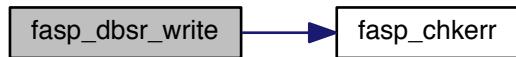
Shiquan Zhang

Date

10/29/2010

Definition at line 1202 of file io.c.

Here is the call graph for this function:



9.45.2.4 void fasp_dbsr_write_coo (const char * *filename*, const **dBSRmat** * *A*)

Print out a **dBSRmat** matrix in coordinate format for matlab spy.

Parameters

<i>filename</i>	Name of file to write to
<i>A</i>	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 1482 of file io.c.

Here is the call graph for this function:



9.45.2.5 void fasp_dcoo1_read (const char * *filename*, dCOOmat * *A*)

Read A from matrix disk file in IJ format – indices starting from 0.

Parameters

<i>filename</i>	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

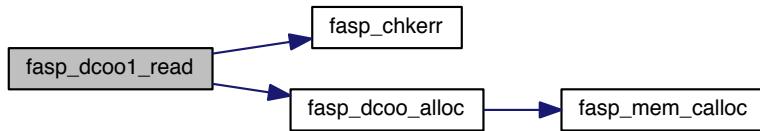
Xiaozhe Hu

Date

03/24/2013

Definition at line 369 of file io.c.

Here is the call graph for this function:



9.45.2.6 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 1423 of file io.c.

9.45.2.7 void fasp_dcoo_read (const char * *filename*, dCSRmat * *A*)

Read *A* from matrix disk file in IJ format – indices starting from 0.

Parameters

<i>filename</i>	File name for matrix
<i>A</i>	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

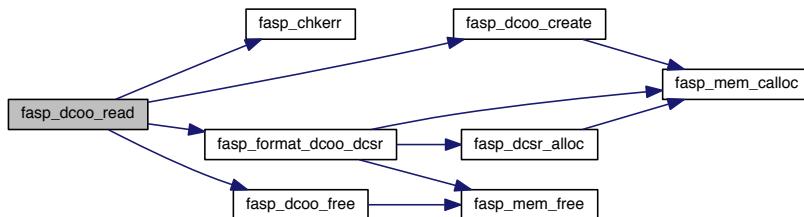
Xuehai Huang, Chensong Zhang

Date

03/29/2009

Definition at line 318 of file io.c.

Here is the call graph for this function:



9.45.2.8 void fasp_dcoo_shift_read (const char * *filename*, dCSRmat * *A*)

Read A from matrix disk file in IJ format – indices starting from 0.

Parameters

<i>filename</i>	File name for matrix
<i>A</i>	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 fashion and converts the matrix to [dCSRmat](#) format.

Author

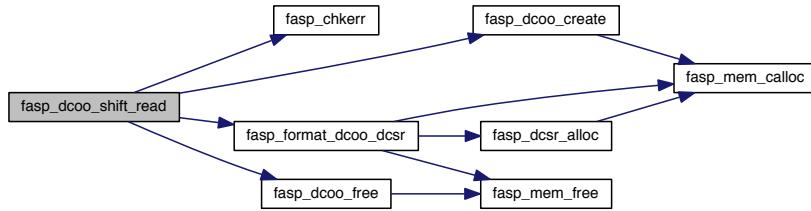
Xiaozhe Hu

Date

04/01/2014

Definition at line 420 of file io.c.

Here is the call graph for this function:



9.45.2.9 void fasp_dcoo_write (const char * *filename*, dCSRmat * *A*)

Write a matrix to disk file in IJ format (coordinate format)

Parameters

<i>A</i>	pointer to the dCSRmat matrix
<i>filename</i>	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 1102 of file io.c.

Here is the call graph for this function:



9.45.2.10 void fasp_dcsr_print(dCSRmat * A)

Print out a `dCSRmat` matrix in coordinate format.

Parameters

<code>A</code>	Pointer to the <code>dCSRmat</code> matrix A
----------------	--

Author

Xuehai Huang

Date

03/29/2009

Definition at line 1401 of file io.c.

9.45.2.11 void fasp_dcsr_read(const char * filename, dCSRmat * A)

Read A from matrix disk file in IJ format.

Parameters

<code>*filename</code>	char for matrix file name
<code>*A</code>	pointer to the CSR matrix

Author

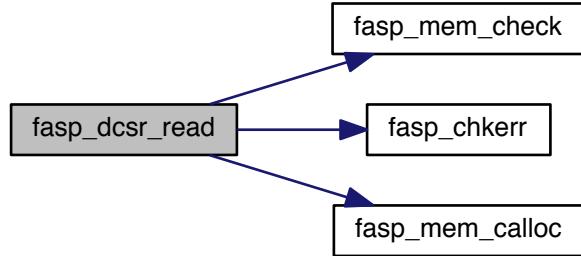
Ziteng Wang

Date

12/25/2012

Definition at line 257 of file io.c.

Here is the call graph for this function:



9.45.2.12 void fasp_dcsr_write_coo (const char * filename, const dCSRmat * A)

Print out a `dCSRmat` matrix in coordinate format for matlab spy.

Parameters

<code>filename</code>	Name of file to write to
<code>A</code>	Pointer to the <code>dCSRmat</code> matrix <code>A</code>

Author

Chunsheng Feng

Date

11/14/2013

Definition at line 1531 of file `io.c`.

Here is the call graph for this function:



9.45.2.13 void fasp_dcsrvec1_read (const char * filename, dCSRmat * A, dvector * b)

Read `A` and `b` from a SINGLE disk file.

Parameters

<i>filename</i>	File name
<i>A</i>	Pointer to the CSR matrix
<i>b</i>	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

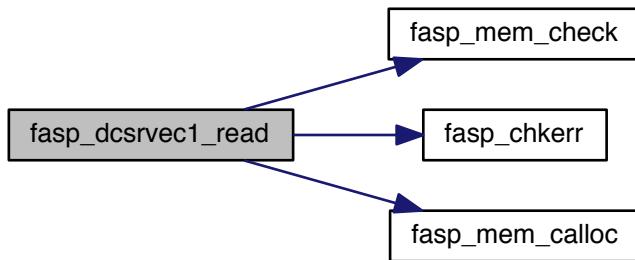
Date

03/29/2009

Modified by Chensong Zhang on 03/14/2012

Definition at line 86 of file `io.c`.

Here is the call graph for this function:



9.45.2.14 `void fasp_dcsrvec1_write(const char * filename, dCSRmat * A, dvector * b)`

Write A and b to a SINGLE disk file.

Parameters

<i>filename</i>	File name
<i>A</i>	Pointer to the CSR matrix
<i>b</i>	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 953 of file io.c.

Here is the call graph for this function:



9.45.2.15 void fasp_dcsrvec2_read (const char * *filemat*, const char * *filerhs*, dCSRmat * *A*, dvector * *b*)

Read A and b from two disk files.

Parameters

<i>filemat</i>	File name for matrix
<i>filerhs</i>	File name for right-hand side
<i>A</i>	Pointer to the dCSR matrix
<i>b</i>	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

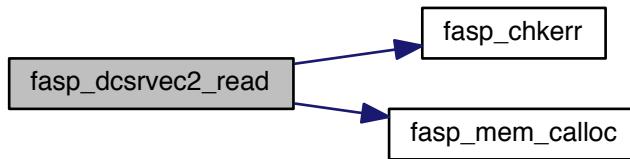
Date

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.

Here is the call graph for this function:



9.45.2.16 void fasp_dcsrvec2_write (const char * *filemat*, const char * *filerhs*, dCSRmat * *A*, dvector * *b*)

Write A and b to two disk files.

Parameters

<i>filemat</i>	File name for matrix
<i>filerhs</i>	File name for right-hand side
<i>A</i>	Pointer to the dCSR matrix
<i>b</i>	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 1031 of file io.c.

Here is the call graph for this function:



9.45.2.17 void fasp_dmtx_read (const char * *filename*, dCSRmat * *A*)

Read A from matrix disk file in MatrixMarket general format.

Parameters

<i>filename</i>	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 **dCSRmat** format. For details of mtx format, please refer to <http://math.nist.gov/MatrixMarket/>. Indices start from 1, NOT 0!!!

Author

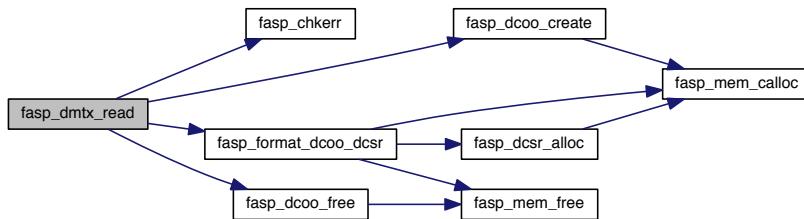
Chensong Zhang

Date

09/05/2011

Definition at line 472 of file io.c.

Here is the call graph for this function:



9.45.2.18 void fasp_dmtxsym_read (const char * *filename*, dCSRmat * A)

Read A from matrix disk file in MatrixMarket sym format.

Parameters

<i>filename</i>	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

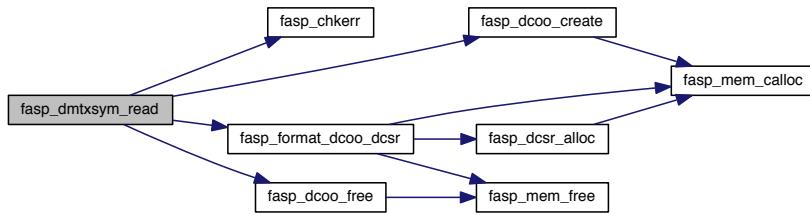
Chensong Zhang

Date

09/02/2011

Definition at line 534 of file io.c.

Here is the call graph for this function:



9.45.2.19 void fasp_dstr_print (dSTRmat * A)

Print out a `dSTRmat` matrix in coordinate format.

Parameters

<code>A</code>	Pointer to the <code>dSTRmat</code> matrix A
----------------	--

Author

Ziteng Wang

Date

12/24/2012

Definition at line 1571 of file io.c.

9.45.2.20 void fasp_dstr_read (const char * filename, dSTRmat * A)

Read A from a disk file in `dSTRmat` format.

Parameters

<code>filename</code>	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

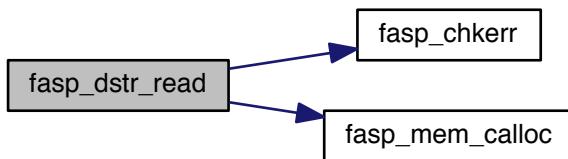
Xuehai Huang

Date

03/29/2009

Definition at line 611 of file io.c.

Here is the call graph for this function:



9.45.2.21 void fasp_dstr_write (const char * *filename*, [dSTRmat](#) * *A*)

Write a [dSTRmat](#) to a disk file.

Parameters

<i>filename</i>	File name for A
A	Pointer to the <code>dSTRmat</code> 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 1142 of file io.c.

Here is the call graph for this function:

**9.45.2.22 void fasp_dvec_print (INT n, dvector * u)**

Print first n entries of a vector of REAL type.

Parameters

<i>n</i>	An interger (if n=0, then print all entries)
<i>u</i>	Pointer to a dvector

Author

Chensong Zhang

Date

03/29/2009

Definition at line 1362 of file io.c.

9.45.2.23 void fasp_dvec_read (const char * filename, dvector * b)

Read b from a disk file in array format.

Parameters

<i>filename</i>	File name for vector b
<i>b</i>	Pointer to the dvector b (output)

Note

File Format:

- nrow
- val_j, j=0:nrow-1

Author

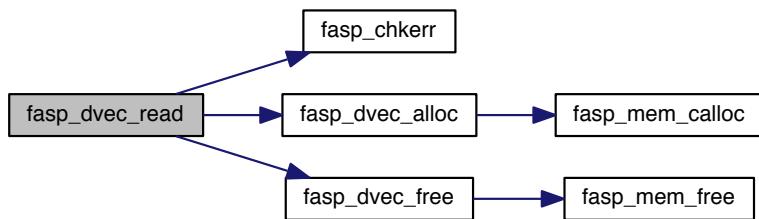
Chensong Zhang

Date

03/29/2009

Definition at line 810 of file io.c.

Here is the call graph for this function:



9.45.2.24 void fasp_dvec_write (const char * *filename*, dvector * *vec*)

Write a dvector to disk file.

Parameters

<i>vec</i>	Pointer to the dvector
<i>filename</i>	File name

Author

Xuehai Huang

Date

03/29/2009

Definition at line 1257 of file io.c.

Here is the call graph for this function:

**9.45.2.25 void fasp_dvecind_read (const char * *filename*, dvector * *b*)**

Read b from matrix disk file.

Parameters

<i>filename</i>	File name for vector b
<i>b</i>	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

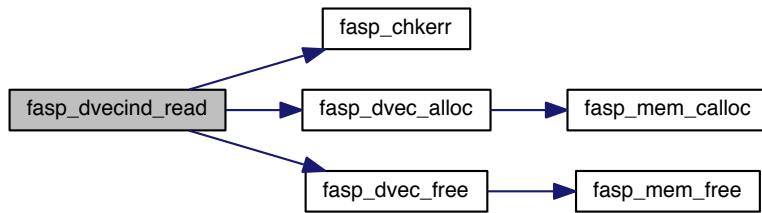
Chensong Zhang

Date

03/29/2009

Definition at line 760 of file io.c.

Here is the call graph for this function:



9.45.2.26 void `fasp_dvecind_write` (const char * *filename*, dvector * *vec*)

Write a dvector to disk file in coordinate format.

Parameters

<i>vec</i>	Pointer to the dvector
<i>filename</i>	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 1293 of file io.c.

Here is the call graph for this function:

**9.45.2.27 fasp_hb_read (*char * input_file*, *dCSRmat * A*, *dvector * b*)**

Read matrix and right-hans side from a HB format file.

Parameters

<i>input_file</i>	File name of vector file
<i>A</i>	Pointer to the matrix
<i>b</i>	Pointer to the vector

Note

Modified from the c code hb_io_prb.c by John Burkardt

Author

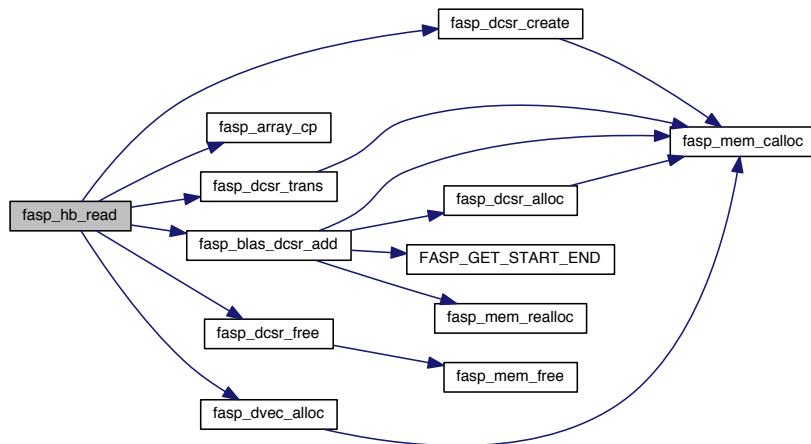
Xiaohe Hu

Date

05/30/2014

Definition at line 2062 of file io.c.

Here is the call graph for this function:



9.45.2.28 void fasp_ivec_print (INT n, ivec * u)

Print first n entries of a vector of INT type.

Parameters

n	An integer (if $n=0$, then print all entries)
u	Pointer to an ivector

Author

Chensong Zhang

Date

03/29/2009

Definition at line 1382 of file io.c.

9.45.3

Read b from a disk file in array format.

9.45.2.29 void fasp_ivec_read (const char * *filename*, ivecotor * *b*)

Read *b* from a disk file in array format.

Parameters

<i>filename</i>	File name for vector b
<i>b</i>	Pointer to the dvector b (output)

Note

File Format:

- nrow
- val_j, j=0:nrow-1

Author

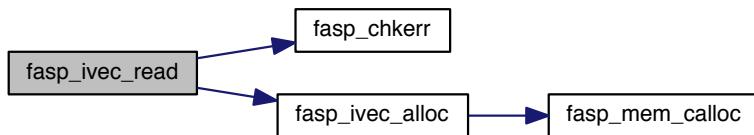
Xuehai Huang

Date

03/29/2009

Definition at line 902 of file io.c.

Here is the call graph for this function:



9.45.2.30 void fasp_ivec_write (const char * *filename*, ivector * *vec*)

Write a ivector to disk file in coordinate format.

Parameters

<i>vec</i>	Pointer to the dvector
<i>filename</i>	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 1328 of file io.c.

Here is the call graph for this function:

**9.45.2.31 void fasp_ivecind_read (const char * *filename*, ivector * *b*)**

Read b from matrix disk file.

Parameters

<i>filename</i>	File name for vector b
<i>b</i>	Pointer to the dvector b (output)

Note

File Format:

- nrow
- ind_j, val_j ... j=0:nrow-1

Author

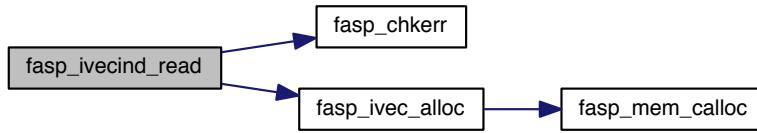
Chensong Zhang

Date

03/29/2009

Definition at line 862 of file io.c.

Here is the call graph for this function:



9.45.2.32 fasp_matrix_read (const char * *filemat*, void * *A*)

Read matrix from different kinds of formats from both ASCII and binary files.

Parameters

<i>filemat</i>	File name of matrix file
<i>A</i>	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

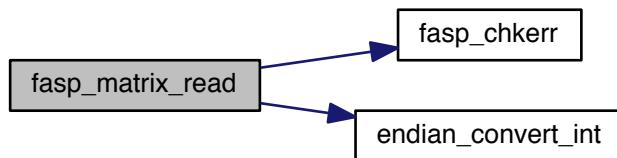
Date

12/24/2012

Modified by Chensong Zhang on 05/01/2013

Definition at line 1605 of file io.c.

Here is the call graph for this function:

**9.45.2.33 void fasp_matrix_read_bin (const char * *filemat*, void * *A*)**

Read matrix in binary format.

Parameters

<i>filemat</i>	File name of matrix file
<i>A</i>	Pointer to the matrix

Author

Xiaozhe Hu

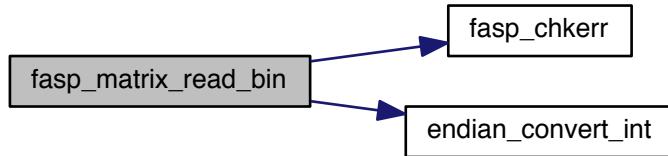
Date

04/14/2013

Modified by Chensong Zhang on 05/01/2013: Use it to read binary files!!!

Definition at line 1710 of file io.c.

Here is the call graph for this function:



9.45.2.34 `fasp_matrix_write(const char *filemat, void *A, INT flag)`

write matrix from different kinds of formats from both ASCII and binary files

Parameters

<code>filemat</code>	File name of matrix file
<code>A</code>	Pointer to the matrix
<code>flag</code>	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

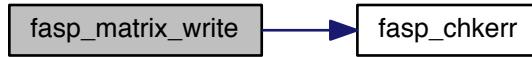
Ziteng Wang

Date

12/24/2012

Definition at line 1784 of file io.c.

Here is the call graph for this function:



9.45.2.35 fasp_vector_read (const char * filerhs, void * b)

Read RHS vector from different kinds of formats from both ASCII and binary files.

Parameters

<i>filerhs</i>	File name of vector file
<i>b</i>	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
- dlenth % third digit of formatflag, length of REAL

Author

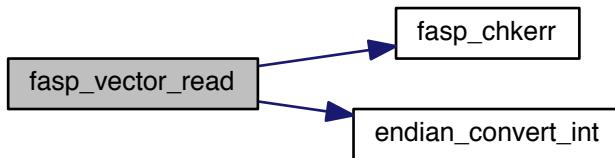
Ziteng Wang

Date

12/24/2012

Definition at line 1877 of file io.c.

Here is the call graph for this function:

**9.45.2.36 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

<i>filerhs</i>	File name of vector file
<i>b</i>	Pointer to the vector
<i>flag</i>	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 1974 of file io.c.

Here is the call graph for this function:



9.45.3 Variable Documentation

9.45.3.1 INT dlength

Length of REAL in byte

Definition at line 14 of file io.c.

9.45.3.2 INT ilength

Length of INT in byte

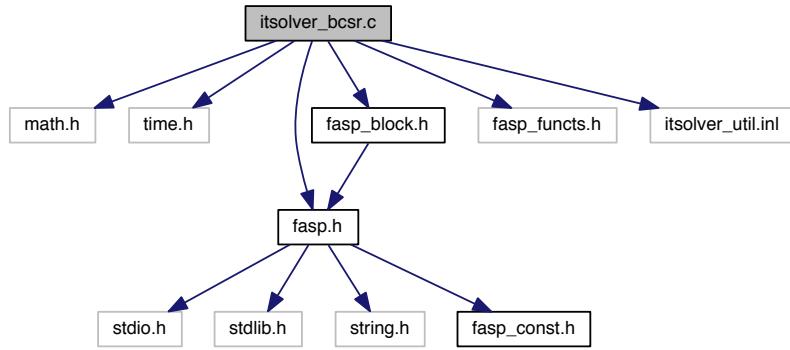
Definition at line 13 of file io.c.

9.46 itsolver_bcsr.c File Reference

Iterative solvers for [block_dCSRmat](#) matrices.

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

Include dependency graph for itsolver_bcsr.c:



Functions

- `INT fasp_solver_bdcsr_itsolver (block_dCSRmat *A, dvector *b, dvector *x, precond *pc, itsolver_param *itparam)`
Solve $Ax = b$ by standard Krylov methods.
- `INT fasp_solver_bdcsr_krylov (block_dCSRmat *A, dvector *b, dvector *x, itsolver_param *itparam)`
Solve $Ax = b$ by standard Krylov methods.
- `INT fasp_solver_bdcsr_krylov_block_3 (block_dCSRmat *A, dvector *b, dvector *x, itsolver_param *itparam, AMG_param *amgparam, dCSRmat *A_diag)`
Solve $Ax = b$ by standard Krylov methods.
- `INT fasp_solver_bdcsr_krylov_block_4 (block_dCSRmat *A, dvector *b, dvector *x, itsolver_param *itparam, AMG_param *amgparam, dCSRmat *A_diag)`
Solve $Ax = b$ by standard Krylov methods.
- `INT fasp_solver_bdcsr_krylov_sweeping (block_dCSRmat *A, dvector *b, dvector *x, itsolver_param *itparam, INT NumLayers, block_dCSRmat *Ai, dCSRmat *local_A, ivector *local_index)`
Solve $Ax = b$ by standard Krylov methods.

9.46.1 Detailed Description

Iterative solvers for `block_dCSRmat` matrices.

9.46.2 Function Documentation

9.46.2.1 INT `fasp_solver_bdcsr_itsolver (block_dCSRmat * A, dvector * b, dvector * x, precond * pc, itsolver_param * itparam)`

Solve $Ax = b$ by standard Krylov methods.

Parameters

<i>A</i>	Pointer to the coeff matrix in block_dCSRmat format
<i>b</i>	Pointer to the right hand side in dvector format
<i>x</i>	Pointer to the approx solution in dvector format
<i>pc</i>	Pointer to the preconditioning action
<i>itparam</i>	Pointer to parameters for iterative solvers

Returns

Number of iterations if succeed

Author

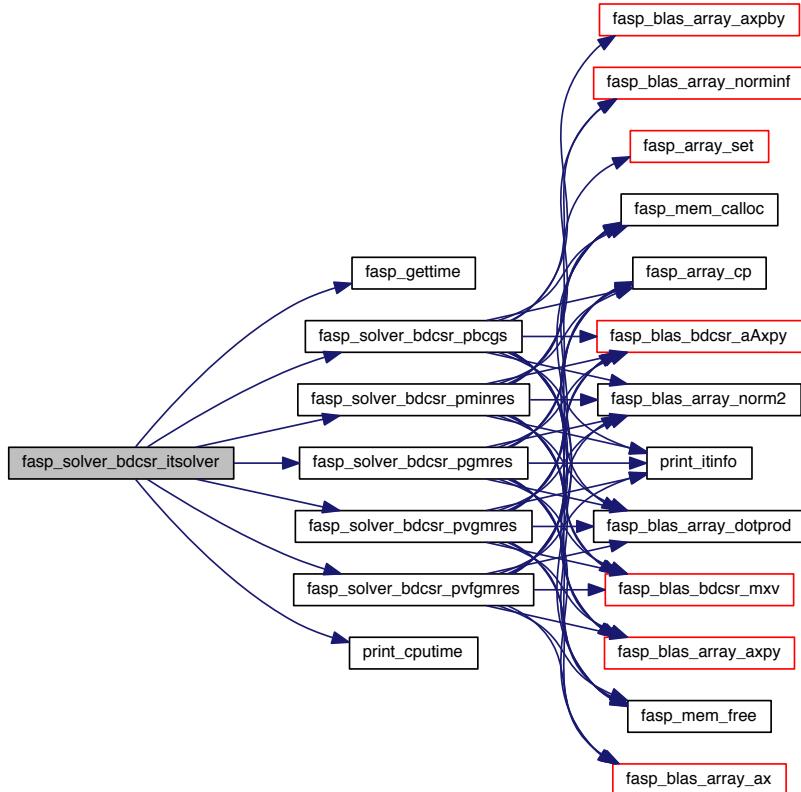
Chensong Zhang

Date

11/25/2010

Definition at line 35 of file `itsolver_bcsr.c`.

Here is the call graph for this function:



9.46.2.2 INT fasp_solver_bdcsr_krylov(block_dCSRmat *A, dvector *b, dvector *x, itsolver_param *itparam)

Solve $Ax = b$ by standard Krylov methods.

Parameters

<i>A</i>	Pointer to the coeff matrix in <code>block_dCSRmat</code> format
<i>b</i>	Pointer to the right hand side in dvector format
<i>x</i>	Pointer to the approx solution in dvector format
<i>itparam</i>	Pointer to parameters for iterative solvers

Returns

Number of iterations if succeed

Author

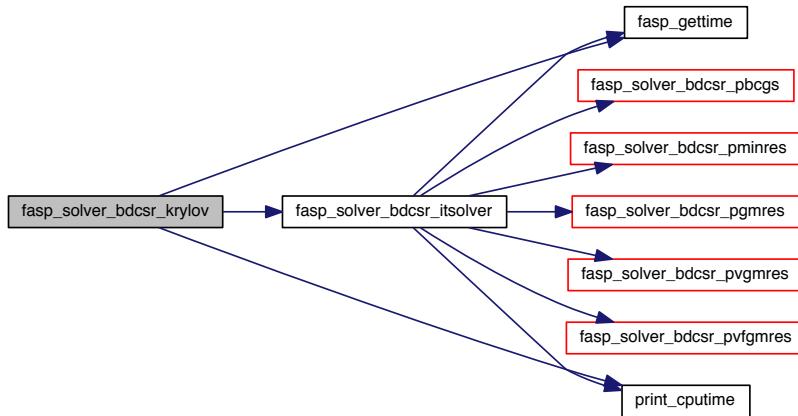
Xiaozhe Hu

Date

07/18/2010

Definition at line 123 of file `itsolver_bcsr.c`.

Here is the call graph for this function:



9.46.2.3 INT `fasp_solver_bdcsr_krylov_block_3` (`block_dCSRmat * A`, `dvector * b`, `dvector * x`, `itsolver_param * itparam`, `AMG_param * amgparam`, `dCSRmat * A_diag`)

Solve $Ax = b$ by standard Krylov methods.

Parameters

<i>A</i>	Pointer to the coeff matrix in block_dCSRmat format
<i>b</i>	Pointer to the right hand side in dvector format
<i>x</i>	Pointer to the approx solution in dvector format
<i>itparam</i>	Pointer to parameters for iterative solvers
<i>amgparam</i>	Pointer to parameters for AMG solvers
<i>A_diag</i>	Digonal blocks of A

Returns

Number of iterations if succeed

Author

Xiaozhe Hu

Date

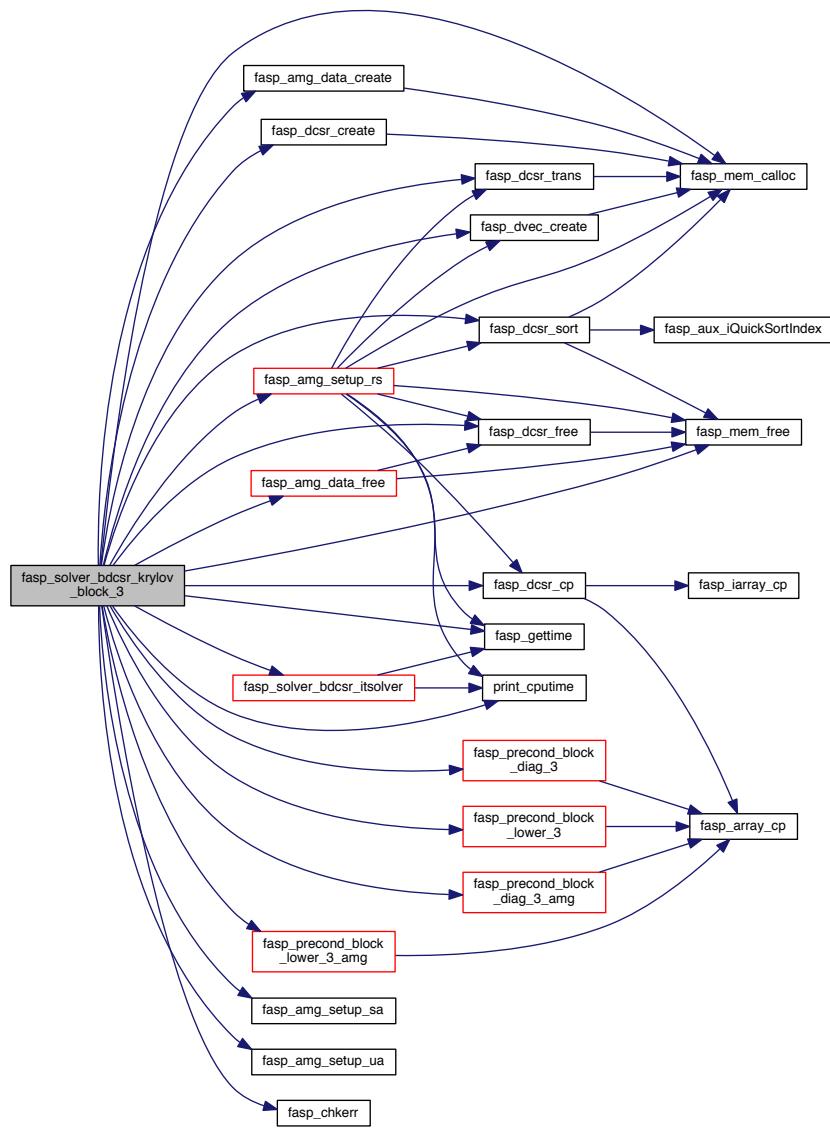
07/10/2014

Note

only works for 3by3 block [dCSRmat](#) problems!! – Xiaozhe Hu

Definition at line 176 of file `itsolver_bcsr.c`.

Here is the call graph for this function:



9.46.2.4 INT fasp_solver_bdcsc_krylov_block_4 (block_dCSRmat * A, dvector * b, dvector * x, itsolver_param * itparam, AMG_param * amgparam, dCSRmat * A_diag)

Solve $Ax = b$ by standard Krylov methods.

Parameters

<i>A</i>	Pointer to the coeff matrix in block_dCSRmat format
<i>b</i>	Pointer to the right hand side in dvector format
<i>x</i>	Pointer to the approx solution in dvector format
<i>itparam</i>	Pointer to parameters for iterative solvers
<i>amgparam</i>	Pointer to parameters for AMG solvers
<i>A_diag</i>	Digonal blocks of A

Returns

Number of iterations if succeed

Author

Xiaozhe Hu

Date

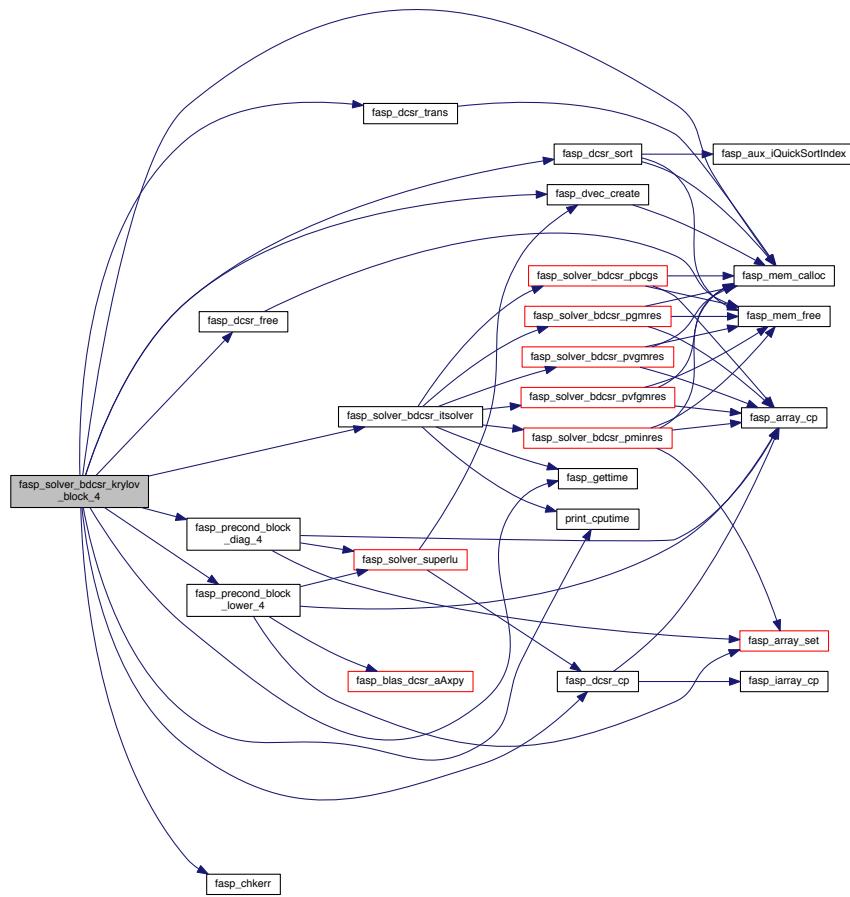
07/06/2014

Note

only works for 4 by 4 block [dCSRmat](#) problems!! – Xiaozhe Hu

Definition at line 365 of file `itsolver_bcsr.c`.

Here is the call graph for this function:



9.46.2.5 INT fasp_solver_bdcsr_krylov_sweeping (*block_dCSRmat* * *A*, *dvector* * *b*, *dvector* * *x*, *itsolver_param* * *itparam*, INT *NumLayers*, *block_dCSRmat* * *Ai*, *dCSRmat* * *local_A*, *ivector* * *local_index*)

Solve $Ax = b$ by standard Krylov methods.

Parameters

<i>A</i>	Pointer to the coeff matrix in <i>block_dCSRmat</i> format
<i>b</i>	Pointer to the right hand side in <i>dvector</i> format
<i>x</i>	Pointer to the approx solution in <i>dvector</i> format
<i>itparam</i>	Pointer to parameters for iterative solvers
<i>NumLayers</i>	Number of layers used for sweeping preconditioner
<i>Ai</i>	Pointer to the coeff matrix for the preconditioner in <i>block_dCSRmat</i> format
<i>local_A</i>	Pointer to the local coeff matrices in the <i>dCSRmat</i> format

<i>local_index</i>	Pointer to the local index in ivector format
--------------------	--

Returns

Number of iterations if succeed

Author

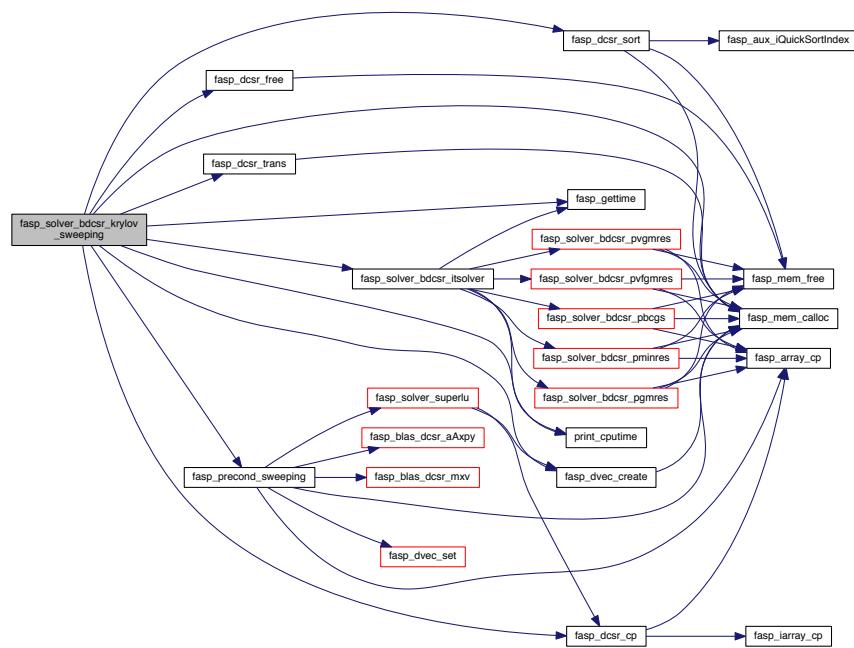
Xiaozhe Hu

Date

05/01/2014

Definition at line 491 of file itsolver_bcsr.c.

Here is the call graph for this function:

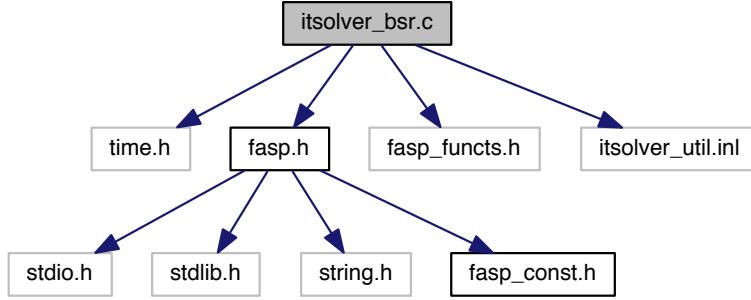


9.47 itsolver_bsr.c File Reference

Iterative solvers for [dBSRmat](#) matrices.

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

Include dependency graph for itsolver_bsr.c:



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 \(dBSRmat *A, dvector *b, dvector **x, itsolver_param *itparam, AMG_param *amgparam, dCSRmat *A_nk, dCSRmat *P_nk, dCSRmat *R_nk\)`](#)
- [`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.

9.47.1 Detailed Description

Iterative solvers for `dBSRmat` matrices.

9.47.2 Function Documentation

9.47.2.1 [`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

<i>A</i>	Pointer to the coeff matrix in <code>dBSRmat</code> format
<i>b</i>	Pointer to the right hand side in dvector format
<i>x</i>	Pointer to the approx solution in dvector format
<i>pc</i>	Pointer to the preconditioning action
<i>itparam</i>	Pointer to parameters for iterative solvers

Returns

Number of iterations if succeed

Author

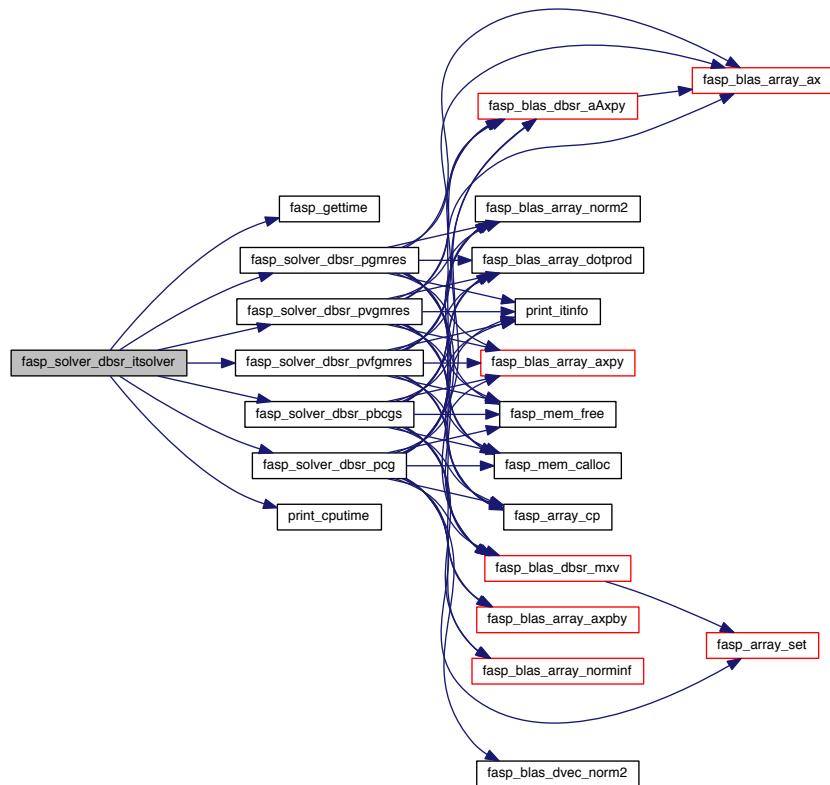
Zhiyang Zhou, Xiaozhe Hu

Date

10/26/2010

Definition at line 37 of file `itsolver_bsr.c`.

Here is the call graph for this function:



9.47.2.2 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

<i>A</i>	Pointer to the coeff matrix in <code>dBSRmat</code> format
<i>b</i>	Pointer to the right hand side in dvector format
<i>x</i>	Pointer to the approx solution in dvector format
<i>itparam</i>	Pointer to parameters for iterative solvers

Returns

Number of iterations if succeed

Author

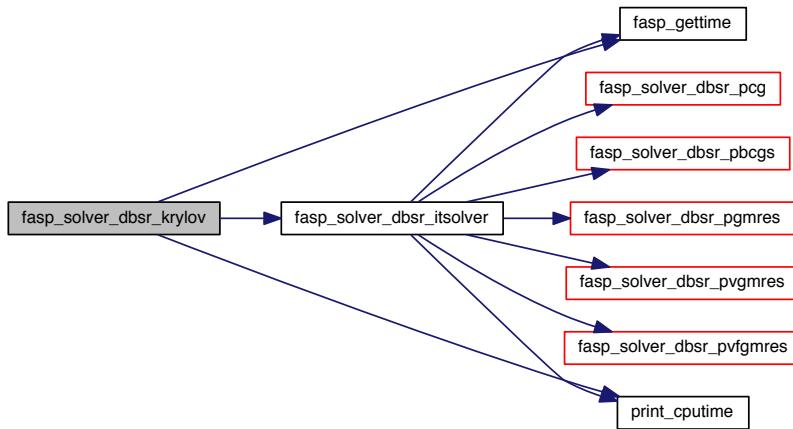
Zhiyang Zhou, Xiaozhe Hu

Date

10/26/2010

Definition at line 126 of file `itsolver_bsr.c`.

Here is the call graph for this function:



9.47.2.3 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

<i>A</i>	Pointer to the coeff matrix in dBSRmat format
<i>b</i>	Pointer to the right hand side in dvector format
<i>x</i>	Pointer to the approx solution in dvector format
<i>itparam</i>	Pointer to parameters for iterative solvers
<i>amgparam</i>	Pointer to parameters of AMG

Returns

Number of iterations if succeed

Author

Xiaozhe Hu

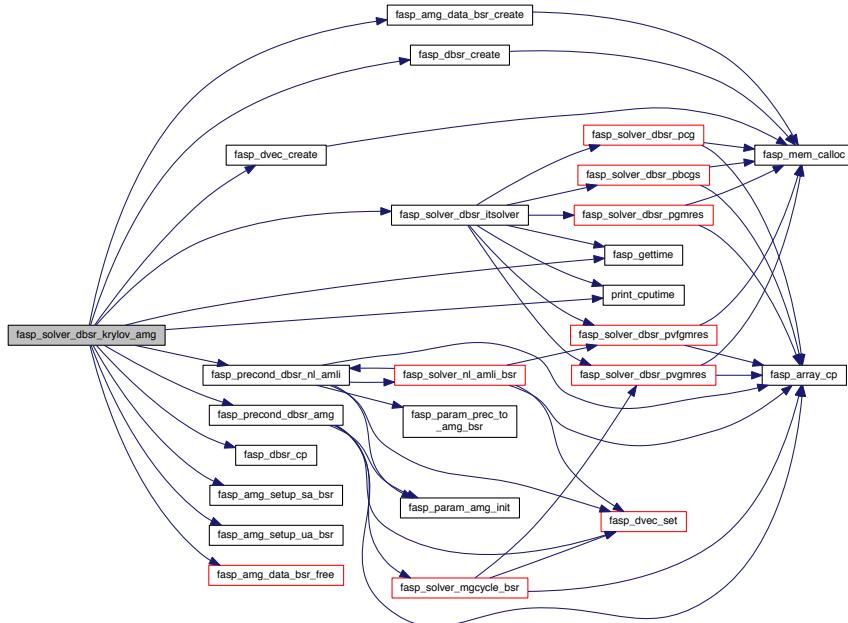
Date

03/16/2012

parameters of iterative method

Definition at line 349 of file `itsolver_bsr.c`.

Here is the call graph for this function:

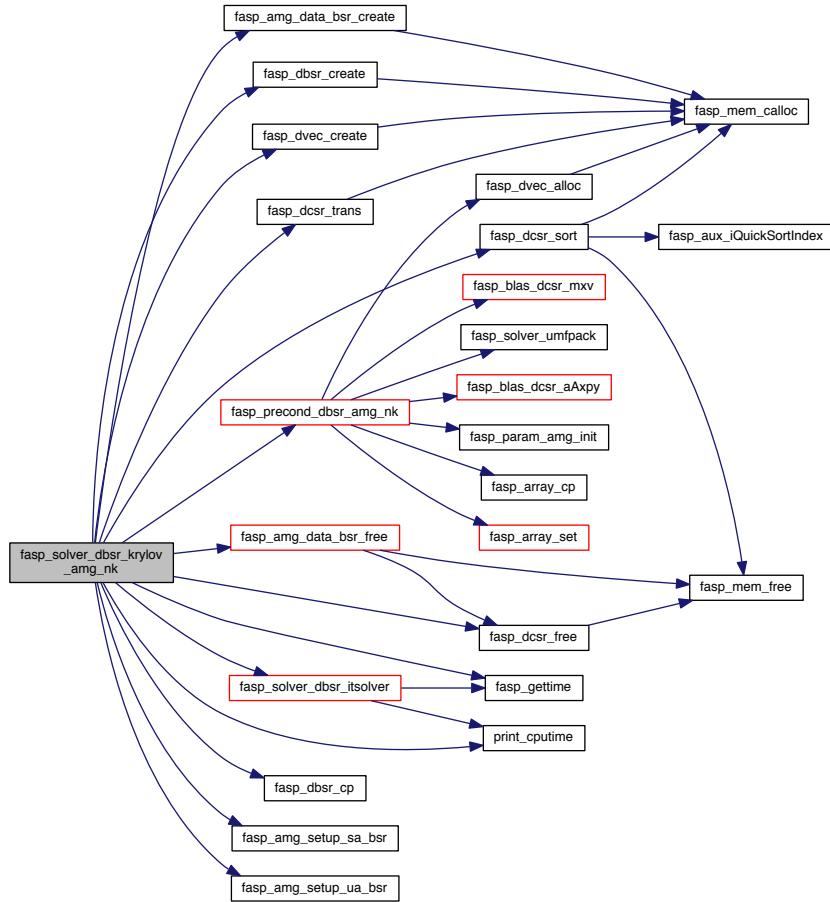


9.47.2.4 INT fasp_solver_dbssr_krylov_amg_nk (**dBSRmat * *A*, **dvector** * *b*, **dvector** * *x*, **itsolver_param** * *itparam*, **AMG_param** * *amgparam*, **dCSRmat** * *A_nk*, **dCSRmat** * *P_nk*, **dCSRmat** * *R_nk*)**

parameters of iterative method

Definition at line 489 of file itsolver_bsr.c.

Here is the call graph for this function:



9.47.2.5 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 <code>dBSRmat</code> format
b	Pointer to the right hand side in <code>dvector</code> format
x	Pointer to the approx solution in <code>dvector</code> format
itparam	Pointer to parameters for iterative solvers

Returns

the number of iterations

Author

Zhiyang Zhou, Xiaozhe Hu

Date

10/26/2010

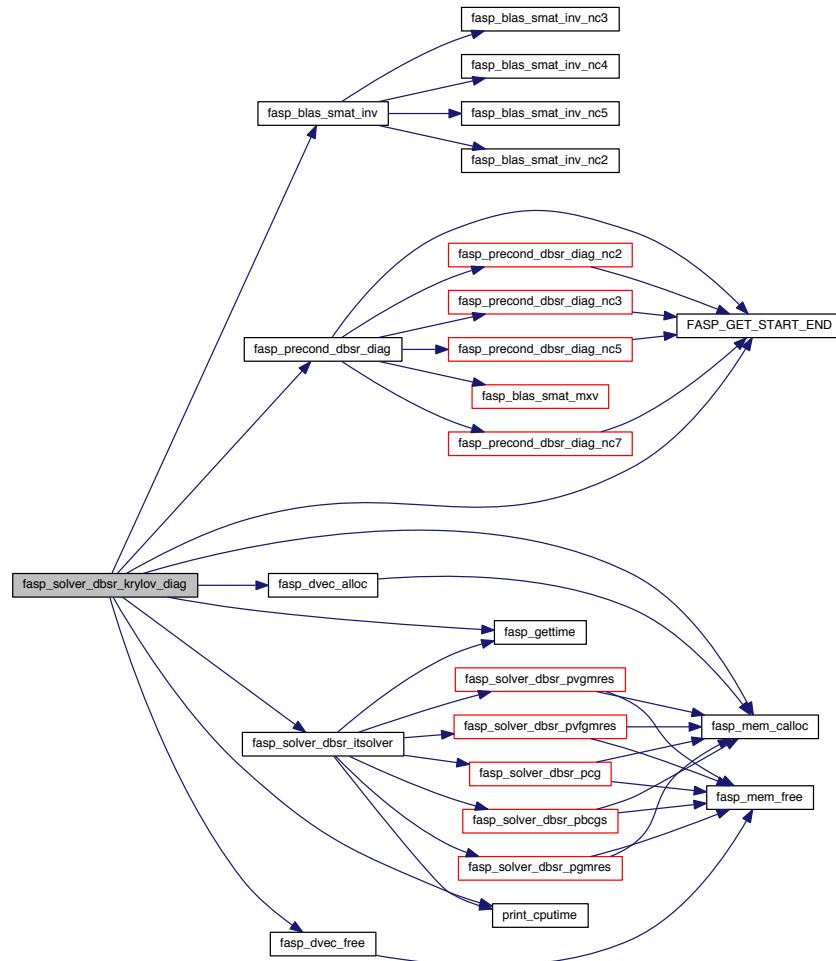
Modified by Chunsheng Feng, Zheng Li

Date

10/15/2012

Definition at line 178 of file itsolver_bsr.c.

Here is the call graph for this function:



9.47.2.6 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

<i>A</i>	Pointer to the coeff matrix in dBSRmat format
<i>b</i>	Pointer to the right hand side in dvector format
<i>x</i>	Pointer to the approx solution in dvector format
<i>itparam</i>	Pointer to parameters for iterative solvers
<i>iluparam</i>	Pointer to parameters of ILU

Returns

Number of iterations if succeed

Author

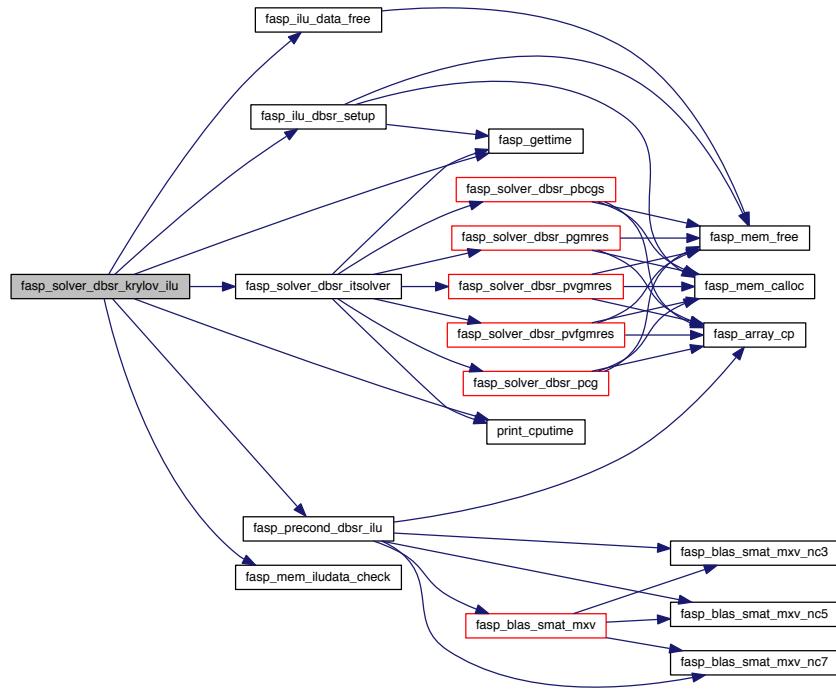
Shiquang Zhang, Xiaozhe Hu

Date

10/26/2010

Definition at line 282 of file `itsolver_bsr.c`.

Here is the call graph for this function:



9.47.2.7 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

<i>A</i>	Pointer to the coeff matrix in dBSRmat format
<i>b</i>	Pointer to the right hand side in dvector format
<i>x</i>	Pointer to the approx solution in dvector format
<i>itparam</i>	Pointer to parameters for iterative solvers
<i>amgparam</i>	Pointer to parameters of AMG
<i>nk_dim</i>	Dimension of the near kernel spaces
<i>nk</i>	Pointer to the near kernal spaces

Returns

Number of iterations if succeed

Author

Xiaozhe Hu

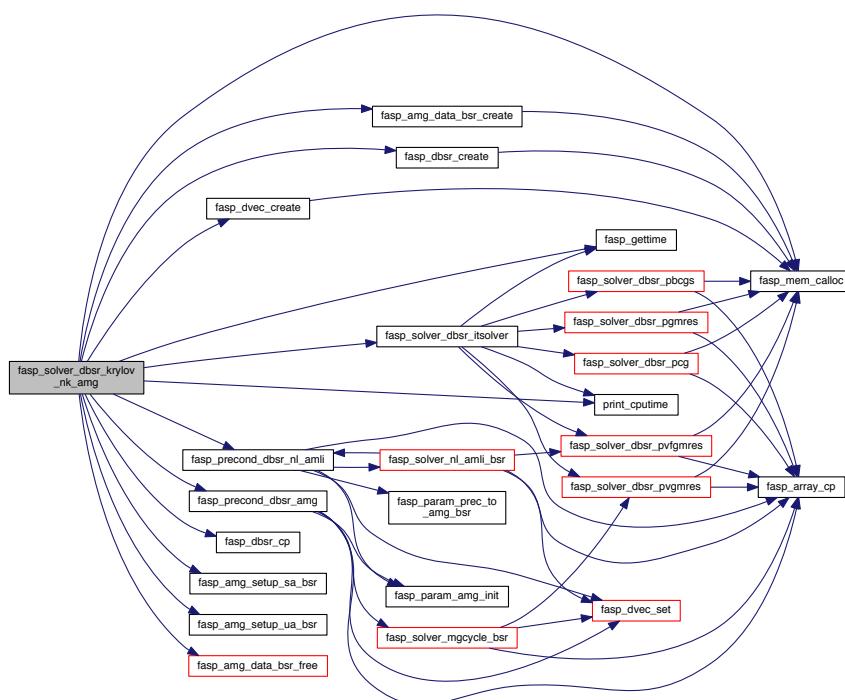
Date

05/27/2012

parameters of iterative method

Definition at line 648 of file `itsolver_bsr.c`.

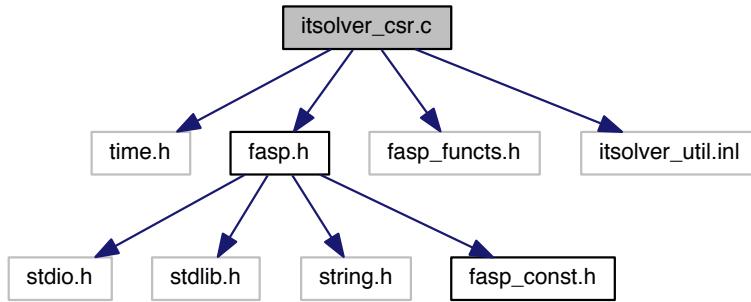
Here is the call graph for this function:



9.48 itsolver_csr.c File Reference

Iterative solvers for [dCSRmat](#) matrices.

```
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
Include dependency graph for itsolver_csr.c:
```



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.

9.48.1 Detailed Description

Iterative solvers for [dCSRmat](#) matrices.

9.48.2 Function Documentation

9.48.2.1 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

<i>A</i>	Pointer to the coeff matrix in dCSRmat format
<i>b</i>	Pointer to the right hand side in dvector format
<i>x</i>	Pointer to the approx solution in dvector format
<i>pc</i>	Pointer to the preconditioning action
<i>itparam</i>	Pointer to parameters for iterative solvers

Returns

Number of iterations if succeed

Author

Chensong Zhang

Date

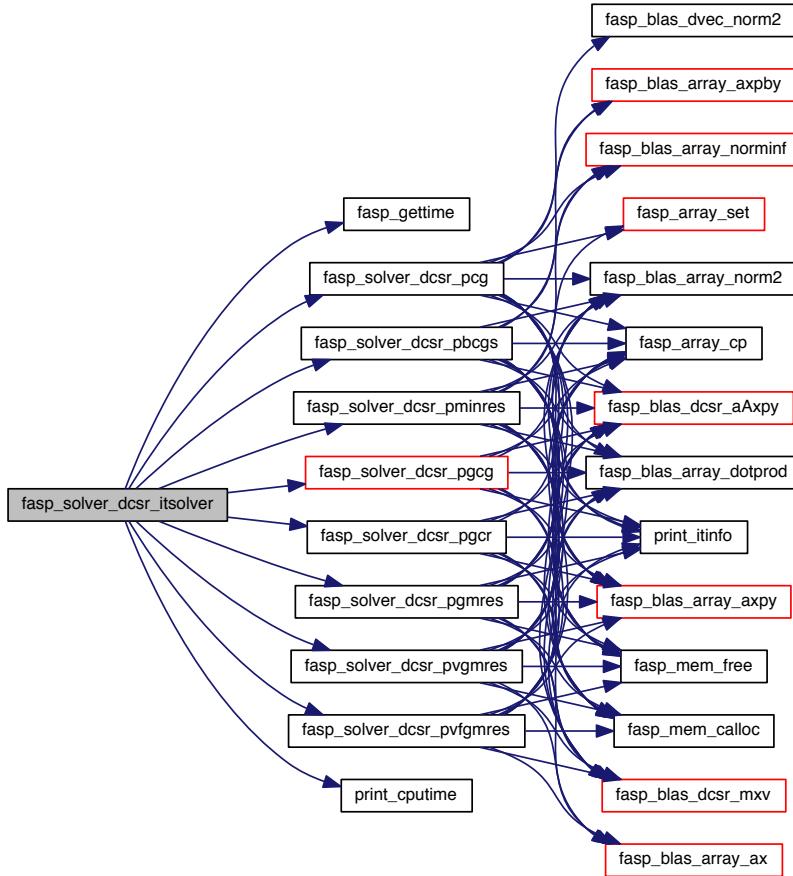
09/25/2009

Note

This is an abstract interface for iterative methods.

Definition at line 39 of file `itsolver_csr.c`.

Here is the call graph for this function:



9.48.2.2 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

<code>A</code>	Pointer to the coeff matrix in <code>dCSRmat</code> format
<code>b</code>	Pointer to the right hand side in <code>dvector</code> format

<i>x</i>	Pointer to the approx solution in dvector format
<i>itparam</i>	Pointer to parameters for iterative solvers

Returns

Number of iterations if succeed

Author

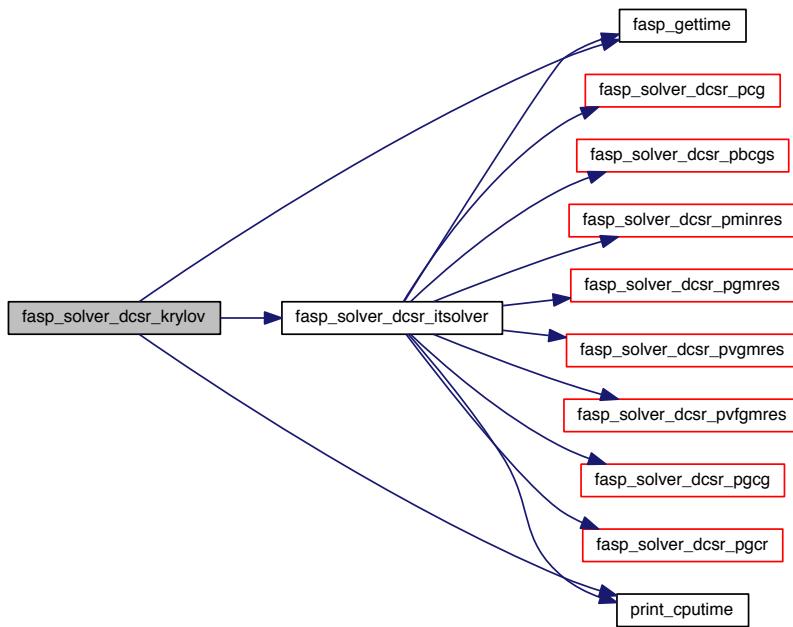
Chensong Zhang, Shiquan Zhang

Date

09/25/2009

Definition at line 143 of file itsolver_csr.c.

Here is the call graph for this function:



9.48.2.3 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

<i>A</i>	Pointer to the coeff matrix in <code>dCSRmat</code> format
<i>b</i>	Pointer to the right hand side in dvector format
<i>x</i>	Pointer to the approx solution in dvector format
<i>itparam</i>	Pointer to parameters for iterative solvers
<i>amgparam</i>	Pointer to parameters for AMG methods

Returns

Number of iterations if succeed

Author

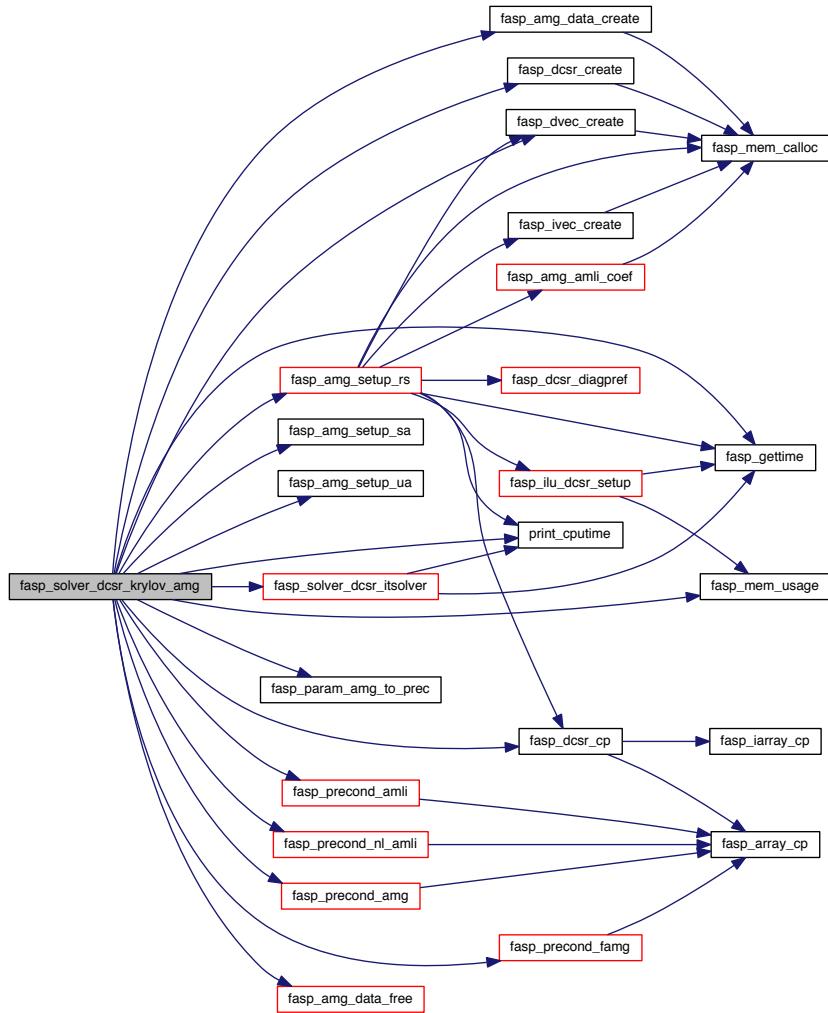
Chensong Zhang

Date

09/25/2009

Definition at line 337 of file `itsolver_csr.c`.

Here is the call graph for this function:



9.48.2.4 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

<i>itparam</i>	Pointer to parameters for iterative solvers
<i>amgparam</i>	Pointer to parameters for AMG methods
<i>A_nk</i>	Pointer to the coeff matrix of near kernel space in dCSRmat format
<i>P_nk</i>	Pointer to the prolongation of near kernel space in dCSRmat format
<i>R_nk</i>	Pointer to the restriction of near kernel space in dCSRmat format

Returns

Number of iterations if succeed

Author

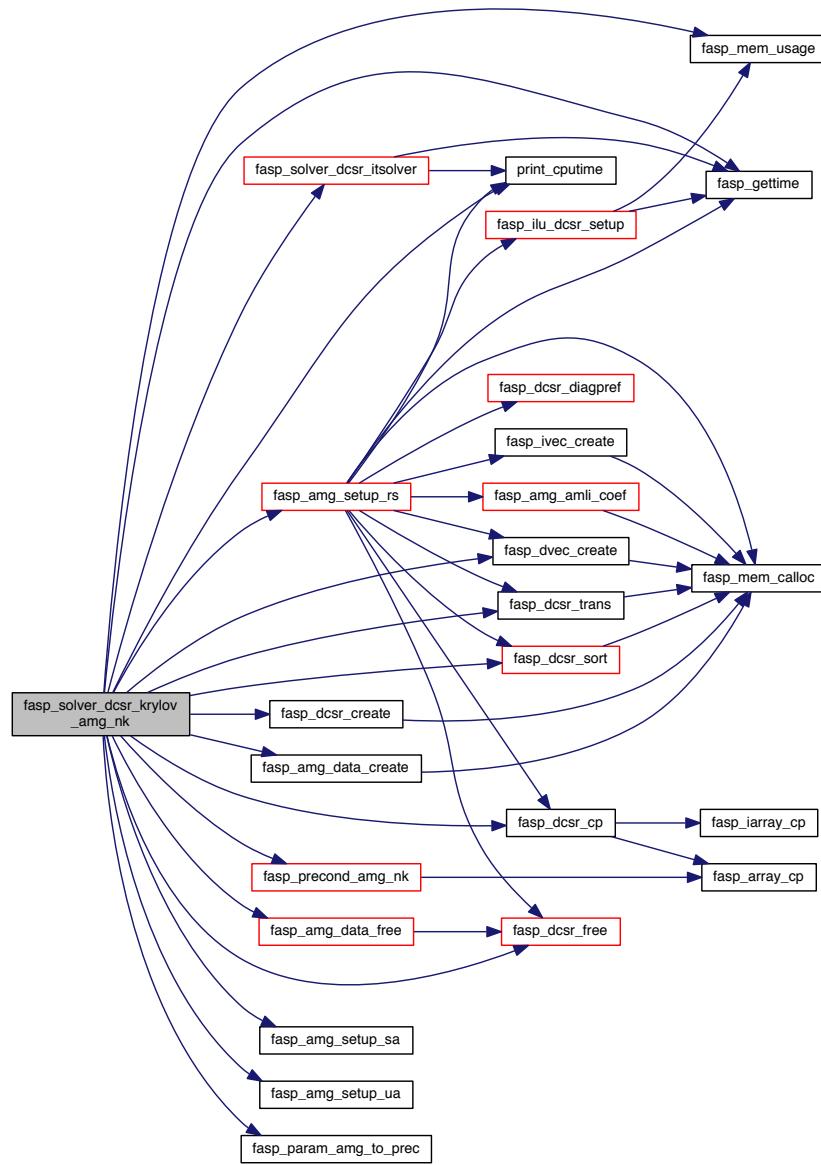
Xiaozhe Hu

Date

05/26/2014

Definition at line 613 of file [itsolver_csr.c](#).

Here is the call graph for this function:



9.48.2.5 INT fasp_solver_dcsr_krylov_diag (dCSRmat * A, dvector * b, dvector * x, itsolver_param * itparam)

Solve $Ax=b$ by diagonal preconditioned Krylov methods.

Parameters

<i>A</i>	Pointer to the coeff matrix in dCSRmat format
<i>b</i>	Pointer to the right hand side in dvector format
<i>x</i>	Pointer to the approx solution in dvector format
<i>itparam</i>	Pointer to parameters for iterative solvers

Returns

Number of iterations if succeed

Author

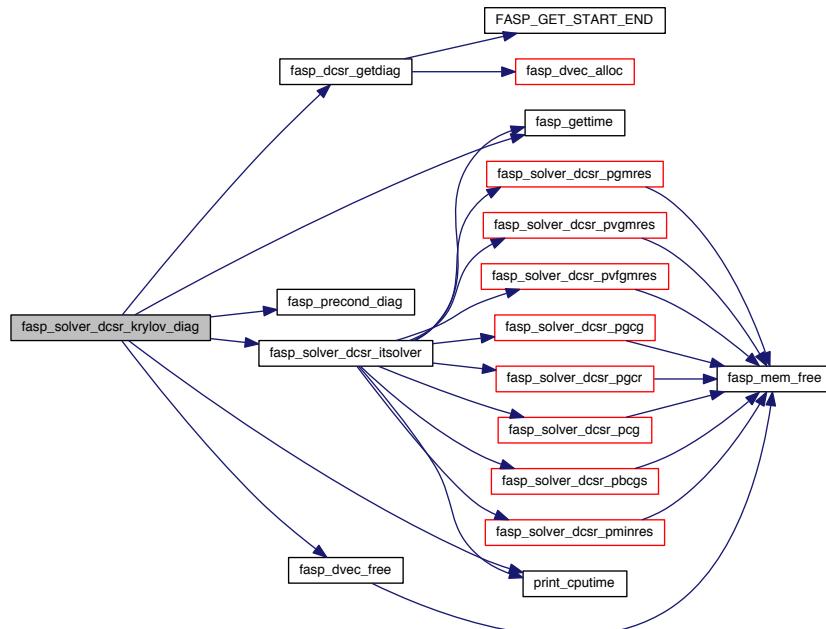
Chensong Zhang, Shiquan Zhang

Date

09/25/2009

Definition at line 193 of file itsolver_csr.c.

Here is the call graph for this function:



9.48.2.6 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

<i>A</i>	Pointer to the coeff matrix in <code>dCSRmat</code> format
<i>b</i>	Pointer to the right hand side in dvector format
<i>x</i>	Pointer to the approx solution in dvector format
<i>itparam</i>	Pointer to parameters for iterative solvers
<i>iluparam</i>	Pointer to parameters for ILU

Returns

Number of iterations if succeed

Author

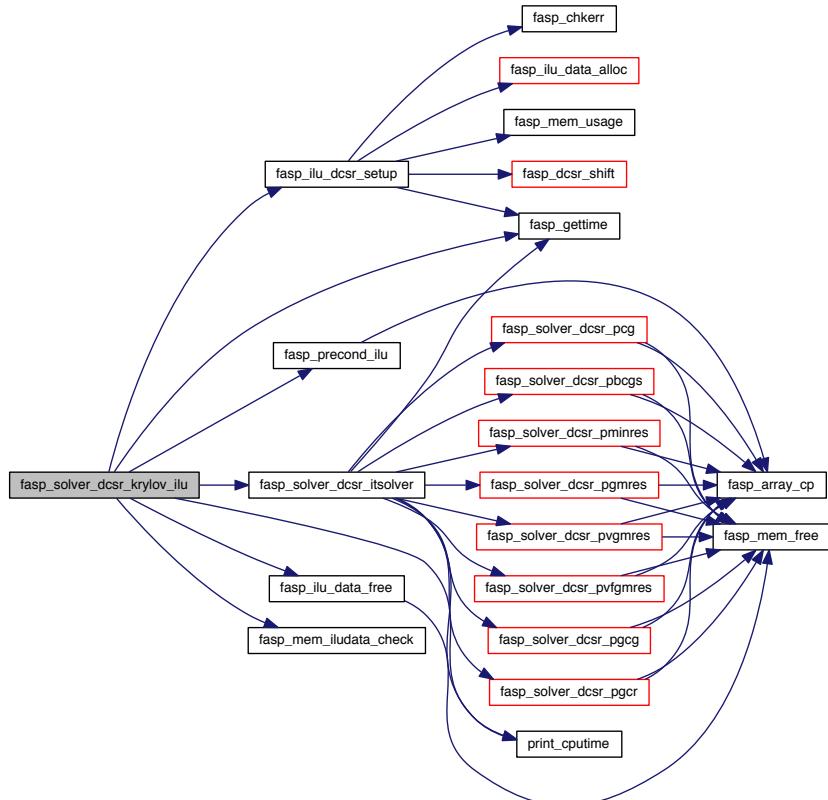
Chensong Zhang, Shiquan Zhang

Date

09/25/2009

Definition at line 442 of file `itsolver_csr.c`.

Here is the call graph for this function:



9.48.2.7 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

<i>A</i>	Pointer to the coeff matrix in <code>dCSRmat</code> format
<i>b</i>	Pointer to the right hand side in dvector format
<i>x</i>	Pointer to the approx solution in dvector format
<i>itparam</i>	Pointer to parameters for iterative solvers
<i>iluparam</i>	Pointer to parameters for ILU
<i>M</i>	Pointer to the preconditioning matrix in <code>dCSRmat</code> format

Returns

Number of iterations if succeed

Author

Xiaozhe Hu

Date

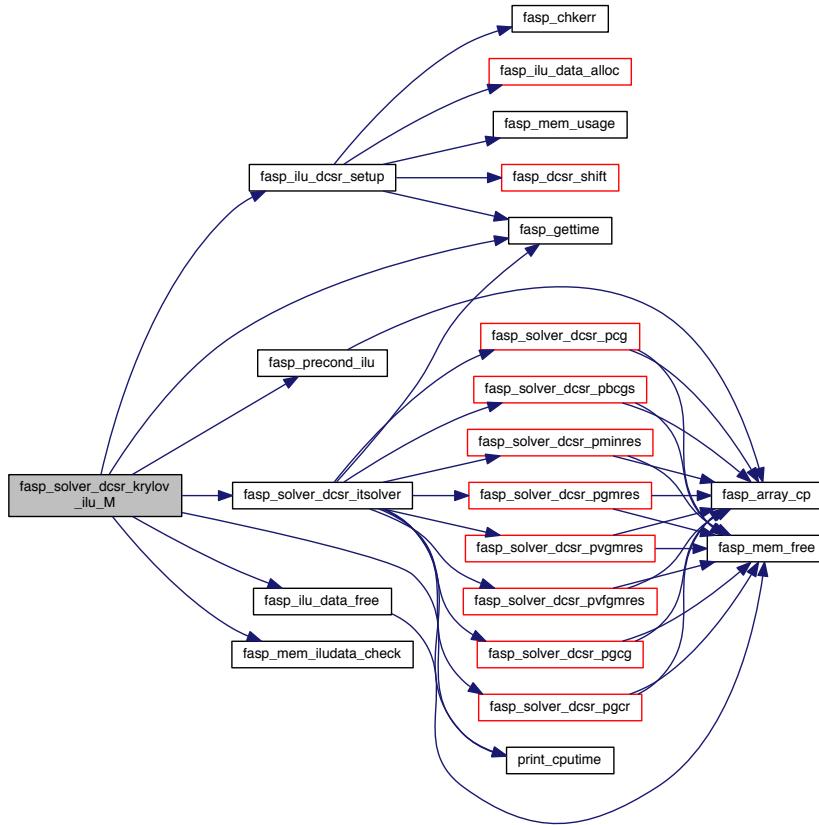
09/25/2009

Note

This function is specially designed for reservoir simulation. Have not been tested in any other places.

Definition at line 529 of file `itsolver_csr.c`.

Here is the call graph for this function:



9.48.2.8 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

<code>A</code>	Pointer to the coeff matrix in <code>dCSRmat</code> format
<code>b</code>	Pointer to the right hand side in <code>dvector</code> format
<code>x</code>	Pointer to the approx solution in <code>dvector</code> format
<code>itparam</code>	Pointer to parameters for iterative solvers
<code>schparam</code>	Pointer to parameters for Schwarz methods

Returns

Number of iterations

Author

Xiaozhe Hu

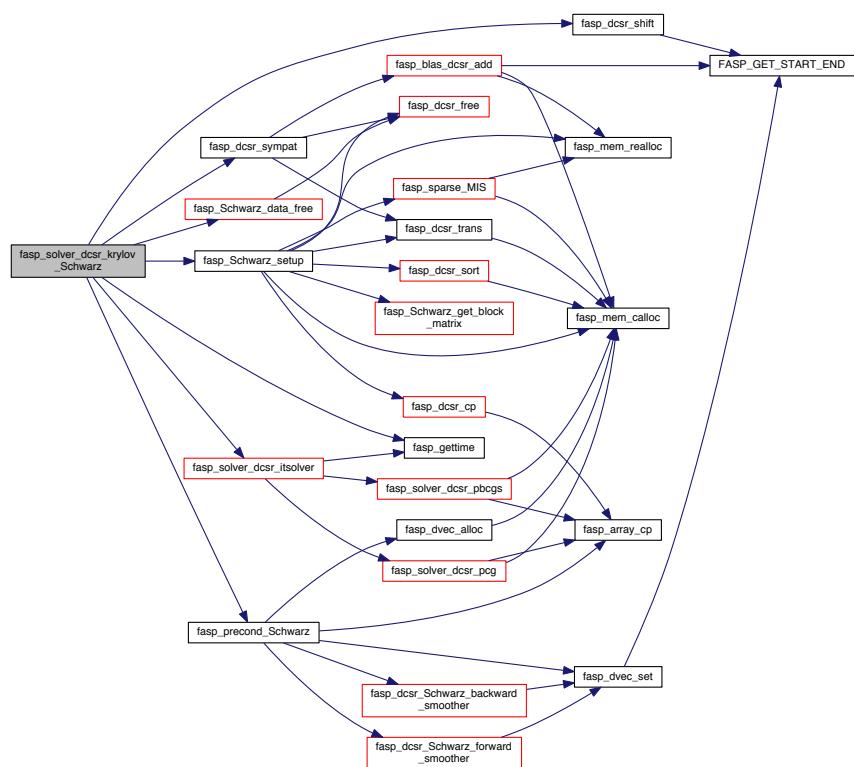
Date

03/21/2011

Modified by Chensong on 07/02/2012: change interface

Definition at line 257 of file itsolver_csr.c.

Here is the call graph for this function:

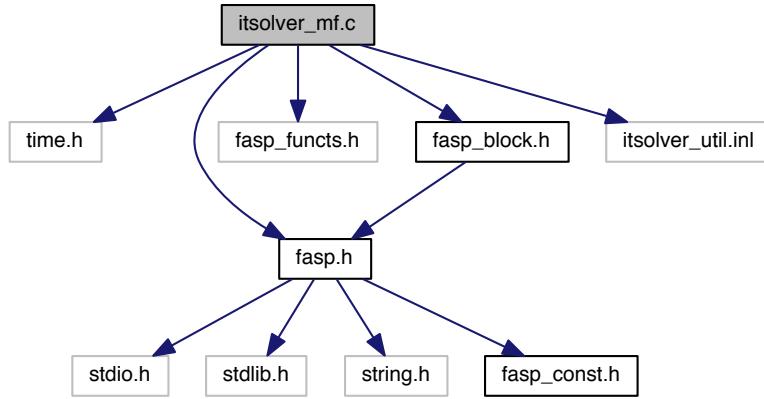


9.49 itsolver_mf.c File Reference

Iterative solvers with matrix-free spmv.

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

Include dependency graph for itsolver_mf.c:



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 itsolvers.

9.49.1 Detailed Description

Iterative solvers with matrix-free spmv.

9.49.2 Function Documentation

9.49.2.1 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

<code>mf</code>	Pointer to <code>mxv_matfree</code> matrix-free spmv operation
<code>b</code>	Pointer to the right hand side in <code>dvector</code> format
<code>x</code>	Pointer to the approx solution in <code>dvector</code> format

<i>pc</i>	Pointer to the preconditioning action
<i>itparam</i>	Pointer to parameters for iterative solvers

Returns

Number of iterations if succeed

Author

Chensong Zhang

Date

09/25/2009

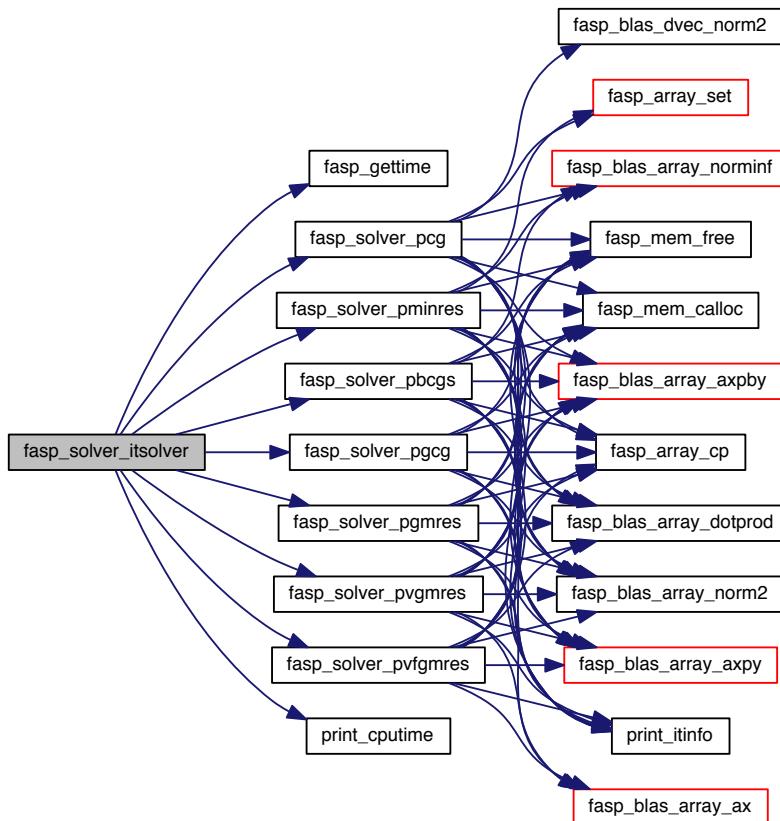
Note

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.

Here is the call graph for this function:



9.49.2.2 void fasp_solver_itsolver_init (INT *matrix_format*, mxv_matfree * *mf*, void * *A*)

Initialize itsolvers.

Parameters

<i>matrix_format</i>	matrix format
<i>mf</i>	Pointer to <code>mxv_matfree</code> matrix-free spmv operation
<i>A</i>	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 198 of file `itsolver_mf.c`.

9.49.2.3 INT fasp_solver_krylov (mxv_matfree * *mf*, dvector * *b*, dvector * *x*, itsolver_param * *itparam*)

Solve Ax=b by standard Krylov methods – without preconditioner.

Parameters

<i>mf</i>	Pointer to <code>mxv_matfree</code> matrix-free spmv operation
<i>b</i>	Pointer to the right hand side in dvector format
<i>x</i>	Pointer to the approx solution in dvector format
<i>itparam</i>	Pointer to parameters for iterative solvers

Returns

Number of iterations if succeed

Author

Chensong Zhang, Shiquan Zhang

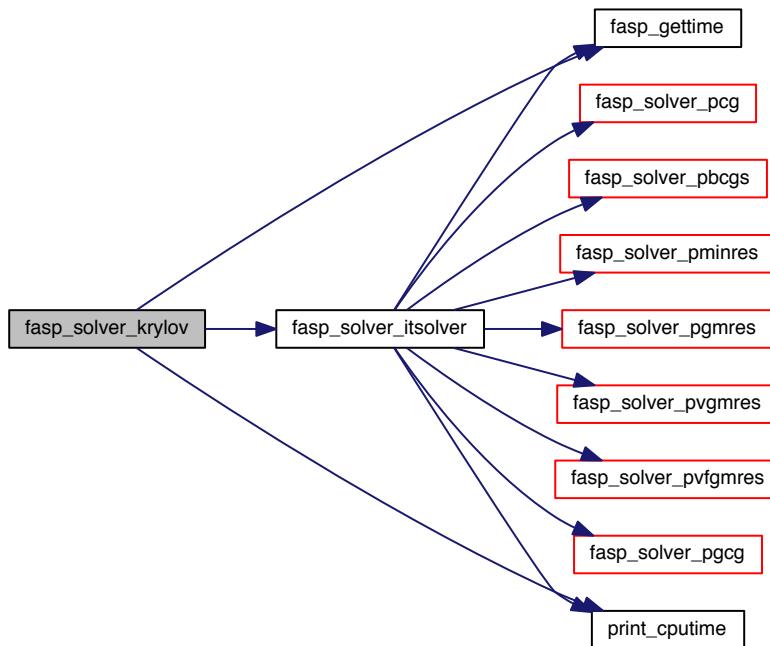
Date

09/25/2009

Modified by Feiteng Huang on 09/20/2012: matrix free

Definition at line 151 of file itsolver_mf.c.

Here is the call graph for this function:

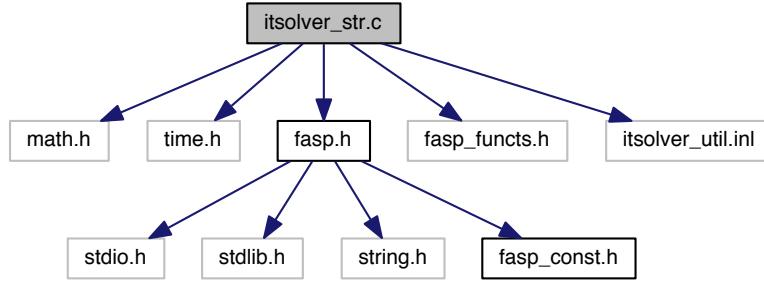


9.50 itsolver_str.c File Reference

Iterative solvers for `dSTRmat` matrices.

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

Include dependency graph for itsolver_str.c:



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, ivec *neigh, ivec *order)`
Solve Ax=b by diagonal preconditioned Krylov methods.

9.50.1 Detailed Description

Iterative solvers for `dSTRmat` matrices.

9.50.2 Function Documentation

9.50.2.1 INT `fasp_solver_dstr_itsolver (dSTRmat * A, dvector * b, dvector * x, precond * pc, itsolver_param * itparam)`

Solve $Ax=b$ by standard Krylov methods.

Parameters

<code>A</code>	Pointer to the coeff matrix in <code>dSTRmat</code> format
----------------	--

<i>b</i>	Pointer to the right hand side in dvector format
<i>x</i>	Pointer to the approx solution in dvector format
<i>pc</i>	Pointer to the preconditioning action
<i>itparam</i>	Pointer to parameters for iterative solvers

Returns

Number of iterations if succeed

Author

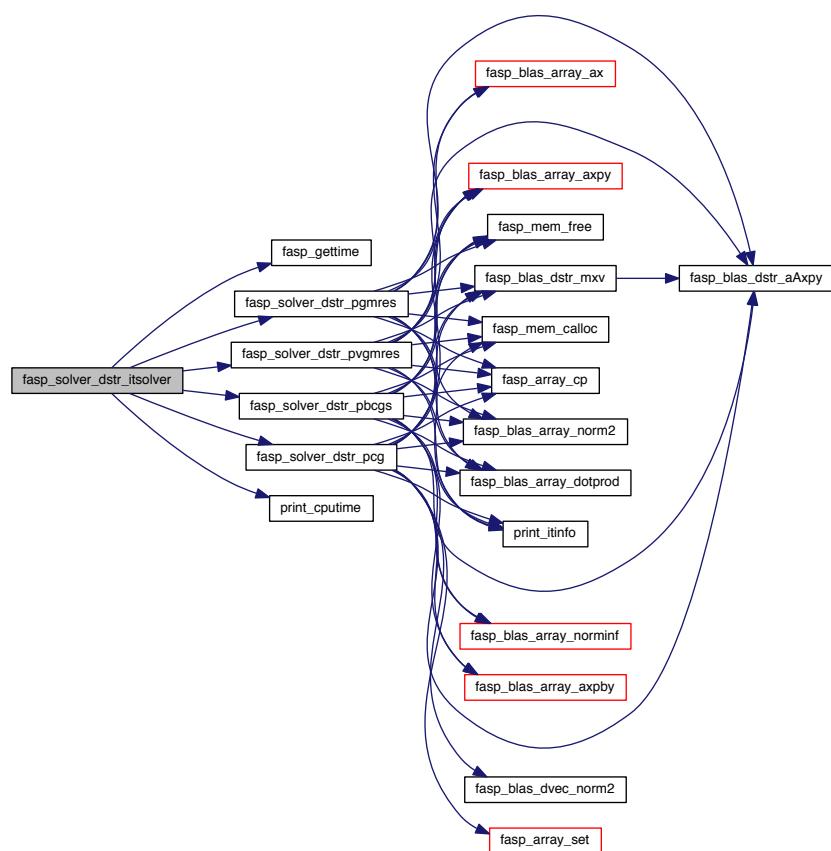
Chensong Zhang

Date

09/25/2009

Definition at line 34 of file itsolver_str.c.

Here is the call graph for this function:



9.50.2.2 INT fasp_solver_dstr_krylov (dSTRmat * *A*, dvector * *b*, dvector * *x*, itsolver_param * *itparam*)

Solve Ax=b by standard Krylov methods.

Parameters

<i>A</i>	Pointer to the coeff matrix in dSTRmat format
<i>b</i>	Pointer to the right hand side in dvector format
<i>x</i>	Pointer to the approx solution in dvector format
<i>itparam</i>	Pointer to parameters for iterative solvers

Returns

Number of iterations if succeed

Author

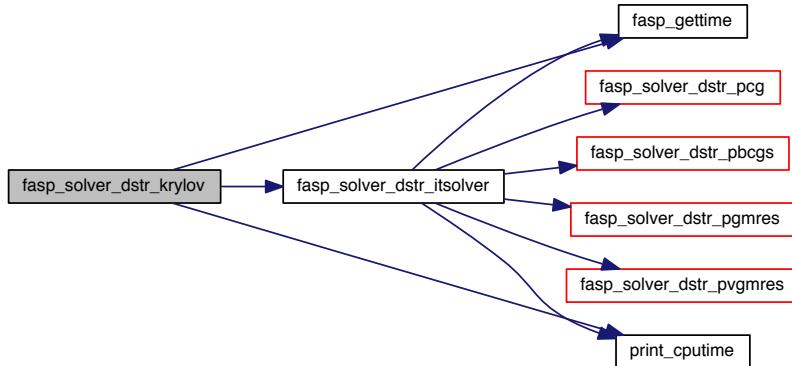
Zhiyang Zhou

Date

04/25/2010

Definition at line 118 of file `itsolver_str.c`.

Here is the call graph for this function:



9.50.2.3 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

<i>A</i>	Pointer to the coeff matrix in dSTRmat format
----------	---

<i>b</i>	Pointer to the right hand side in dvector format
<i>x</i>	Pointer to the approx solution in dvector format
<i>itparam</i>	Pointer to parameters for iterative solvers
<i>neigh</i>	Pointer to neighbor vector
<i>order</i>	Pointer to solver ordering

Returns

Number of iterations if succeed

Author

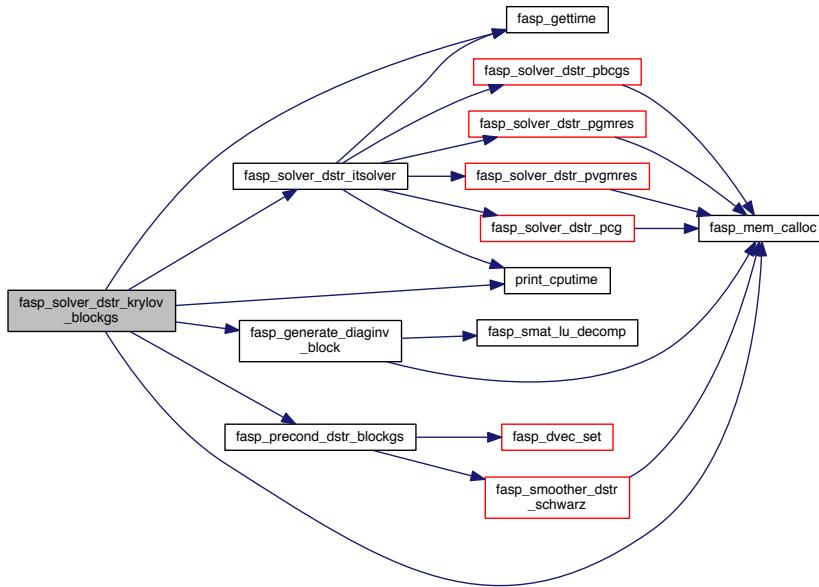
Xiaozhe Hu

Date

10/10/2010

Definition at line 324 of file itsolver_str.c.

Here is the call graph for this function:



9.50.2.4 INT fasp_solver_dstr_krylov_diag (dSTRmat * A, dvector * b, dvector * x, itsolver_param * itparam)

Solve Ax=b by diagonal preconditioned Krylov methods.

Parameters

<i>A</i>	Pointer to the coeff matrix in <code>dSTRmat</code> format
<i>b</i>	Pointer to the right hand side in dvector format
<i>x</i>	Pointer to the approx solution in dvector format
<i>itparam</i>	Pointer to parameters for iterative solvers

Returns

Number of iterations if succeed

Author

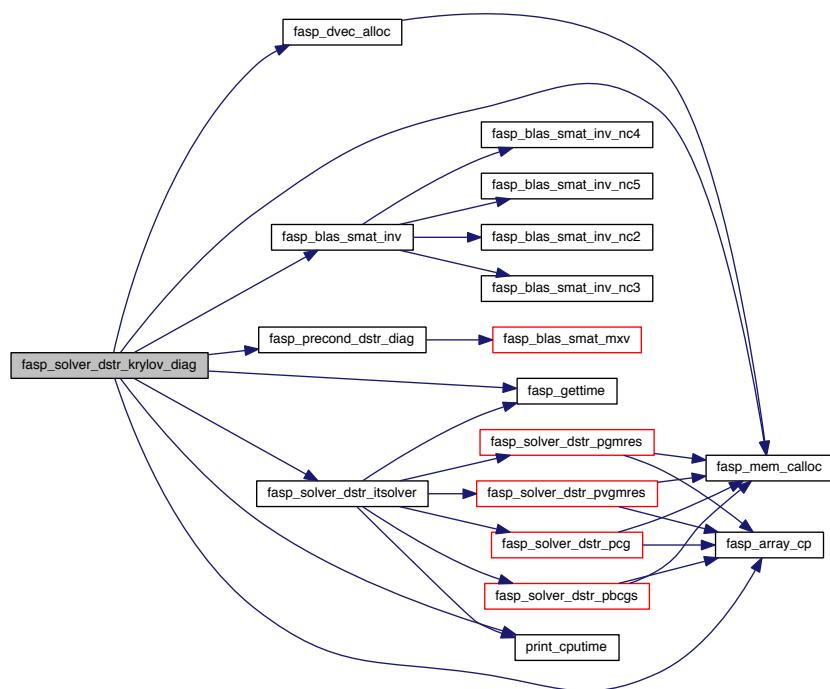
Zhiyang Zhou

Date

4/23/2010

Definition at line 166 of file `itsolver_str.c`.

Here is the call graph for this function:



9.50.2.5 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

<i>A</i>	Pointer to the coeff matrix in <code>dSTRmat</code> format
<i>b</i>	Pointer to the right hand side in dvector format
<i>x</i>	Pointer to the approx solution in dvector format
<i>itparam</i>	Pointer to parameters for iterative solvers
<i>iluparam</i>	Pointer to parameters for ILU

Returns

Number of iterations if succeed

Author

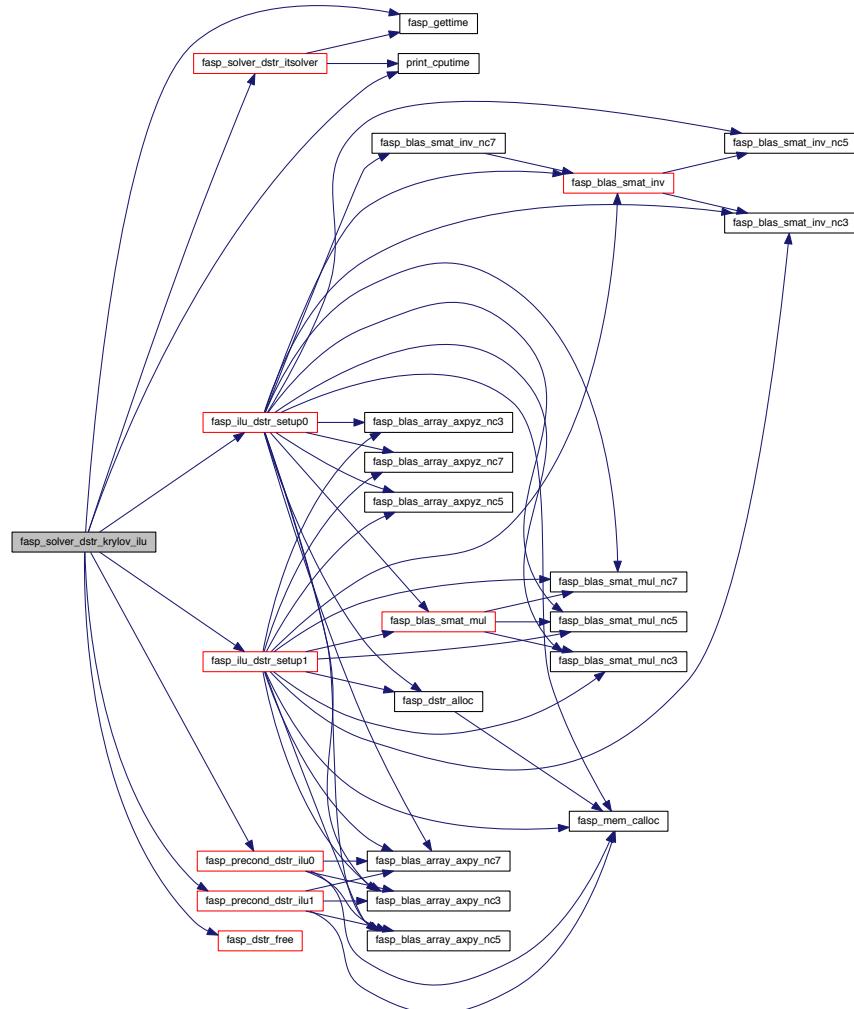
Xiaozhe Hu

Date

05/01/2010

Definition at line 233 of file `itsolver_str.c`.

Here is the call graph for this function:

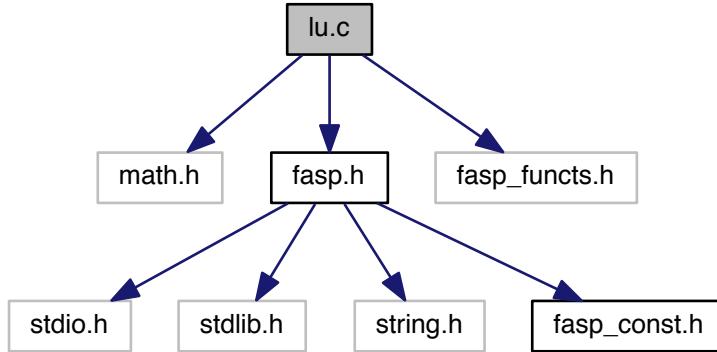


9.51 lu.c File Reference

LU decomposition and direct solve for dense matrix.

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

Include dependency graph for lu.c:



Functions

- **SHORT fasp_smat_lu_decomp (REAL *A, INT pivot[], INT n)**
LU decomposition of A usind Doolittle's method.
- **SHORT fasp_smat_lu_solve (REAL *A, REAL b[], INT pivot[], REAL x[], INT n)**
Solving Ax=b using LU decomposition.

9.51.1 Detailed Description

LU decomposition and direct solve for dense matrix.

9.51.2 Function Documentation

9.51.2.1 **SHORT fasp_smat_lu_decomp (REAL * A, INT pivot[], INT n)**

LU decomposition of A usind Doolittle's method.

Parameters

<i>A</i>	Pointer to the full matrix
<i>pivot</i>	Pivoting positions
<i>n</i>	Size of matrix A

Returns

FASP_SUCCESS if succeed, ERROR_UNKNOWN if fail

Note

Use Doolittle's method to decompose the $n \times n$ 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, $k = 0, \dots, n - 1$ evaluate in order the following pair of expressions $U[k][j] = A[k][j] - (L[k][0]*U[0][j]) + \dots + L[k][k-1]*U[k-1][j])$ for $j = k, k+1, \dots, n-1$ $L[i][k] = (A[i][k] - (L[i][0]*U[0][k]) + \dots + L[i][k-1]*U[k-1][k])) / U[k][k]$ for $i = k+1, \dots, n-1$.

Author

Xuehai Huang

Date

04/02/2009

Definition at line 46 of file lu.c.

9.51.2.2 **SHORT faspmat_lu_solve (REAL * A, REAL b[], INT pivot[], REAL x[], INT n)**

Solving $Ax=b$ using LU decomposition.

Parameters

<i>A</i>	Pointer to the full matrix
<i>b</i>	Right hand side array
<i>pivot</i>	Pivoting positions
<i>x</i>	Pointer to the solution array
<i>n</i>	Size of matrix A

Returns

FASP_SUCCESS if succeed, ERROR_UNKNOWN if failed

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

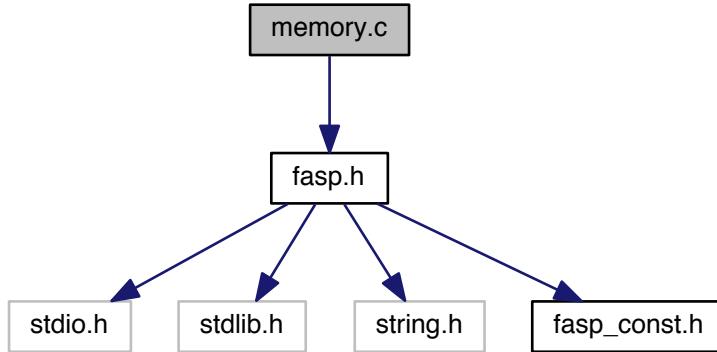
Definition at line 117 of file lu.c.

9.52 memory.c File Reference

Memory allocation and deallocation.

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

Include dependency graph for memory.c:



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 whether a point is null or not.
- **SHORT fasp_mem_iludata_check** (ILU_data *iludata)
Check whether a ILU_data has enough work space.
- **SHORT fasp_mem_dCSRmat_check** (dCSRmat *A)
Check whether a dCSRmat A has successfully 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.

9.52.1 Detailed Description

Memory allocation and deallocation.

9.52.2 Function Documentation

9.52.2.1 void * fasp_mem_calloc (**LONGLONG size, INT type**)

1M = 1024*1024

Allocate, initiate, and check memory

Parameters

<i>size</i>	Number of memory blocks
<i>type</i>	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.

9.52.2.2 SHORT fasp_mem_check (void * *ptr*, const char * *message*, INT *ERR*)

Check whether a point is null or not.

Parameters

<i>ptr</i>	Void pointer to be checked
<i>message</i>	Error message to print
<i>ERR</i>	Integer error code

Returns

FASP_SUCCESS or error code

Author

Chensong Zhang

Date

11/16/2009

Definition at line 192 of file memory.c.

9.52.2.3 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 242 of file `memory.c`.

9.52.2.4 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 145 of file `memory.c`.

9.52.2.5 SHORT fasp_mem_iludata_check(ILU_data * iludata)

Check wether a [ILU_data](#) has enough work space.

Parameters

<i>iludata</i>	Pointer to be checked
----------------	-----------------------

Returns

FASP_SUCCESS if success, else ERROR (negative value)

Author

Xiaozhe Hu, Chensong Zhang

Date

11/27/09

Definition at line 216 of file memory.c.

9.52.2.6 void *fasp_mem_realloc (void *oldmem, LONGLONG type)

Reallocate, initiate, and check memory.

Parameters

<i>oldmem</i>	Pointer to the existing mem block
<i>type</i>	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 111 of file memory.c.

9.52.2.7 void fasp_mem_usage ()

Show total allocated memory currently.

Author

Chensong Zhang

Date

2010/08/12

Definition at line 170 of file memory.c.

9.52.3 Variable Documentation

9.52.3.1 unsigned INT total_alloc_count = 0

Total allocated memory amount.

total allocation times

Definition at line 33 of file memory.c.

9.52.3.2 unsigned INT total_alloc_mem = 0

total allocated memory

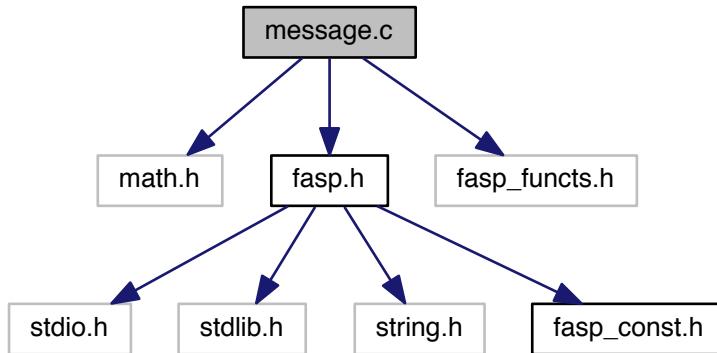
Definition at line 32 of file memory.c.

9.53 message.c File Reference

Output some useful messages.

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

Include dependency graph for message.c:



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 ptrlvl)

Print complexities of AMG method.
- void [print_amgcomplexity_bsr](#) (AMG_data_bsr *mgl, const SHORT ptrlvl)

- 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.

9.53.1 Detailed Description

Output some useful messages.

Note

These routines are meant for internal use only.

9.53.2 Function Documentation

9.53.2.1 void fasp_chkerr (const **SHORT** status, const char * fctname)

Check error status and print out error messages before quit.

Parameters

<i>status</i>	Error status
<i>fctname</i>	Function name where this routine is called

Author

Chensong Zhang

Date

01/10/2012

Definition at line 199 of file message.c.

9.53.2.2 void void print_amgcomplexity (**AMG_data** * mgl, const **SHORT** prtlvl)

Print complexities of AMG method.

Parameters

<i>mgl</i>	Multilevel hierarchy for AMG
<i>prtlvl</i>	How much information to print

Author

Chensong Zhang

Date

11/16/2009

Definition at line 79 of file message.c.

9.53.2.3 void void print_amgcomplexity_bsr (AMG_data_bsr * *mgl*, const SHORT *ptrlvl*)

Print complexities of AMG method for BSR matrices.

Parameters

<i>mgl</i>	Multilevel hierarchy for AMG
<i>ptrlvl</i>	How much information to print

Author

Chensong Zhang

Date

05/10/2013

Definition at line 122 of file message.c.

9.53.2.4 void void print_cputime (const char * *message*, const REAL *cputime*)

Print CPU walltime.

Parameters

<i>message</i>	Some string to print out
<i>cputime</i>	Walltime since start to end

Author

Chensong Zhang

Date

04/10/2012

Definition at line 165 of file message.c.

9.53.2.5 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

<i>ptrlvl</i>	Level for output
<i>stop_type</i>	Type of stopping criteria
<i>iter</i>	Number of iterations
<i>relres</i>	Relative residual of different kinds

<i>absres</i>	Absolute residual of different kinds
<i>factor</i>	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.

9.53.2.6 void print_message (const INT *ptrlvl*, const char * *message*)

Print output information if necessary.

Parameters

<i>ptrlvl</i>	Level for output
<i>message</i>	Error message to print

Author

Chensong Zhang

Date

11/16/2009

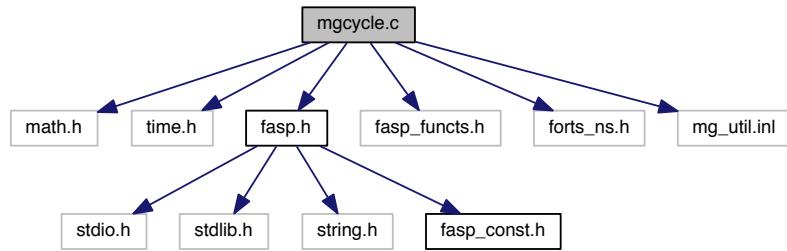
Definition at line 182 of file message.c.

9.54 mgcycle.c File Reference

Abstract non-recursive multigrid cycle.

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

Include dependency graph for mgcycle.c:



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.

9.54.1 Detailed Description

Abstract non-recursive multigrid cycle.

9.54.2 Function Documentation

9.54.2.1 void [fasp_solver_mgcycle](#) ([AMG_data](#) * *mgl*, [AMG_param](#) * *param*)

Solve Ax=b with non-recursive multigrid cycle.

Parameters

<i>mgl</i>	Pointer to AMG data: AMG_data
<i>param</i>	Pointer to AMG parameters: AMG_param

Author

Chensong Zhang

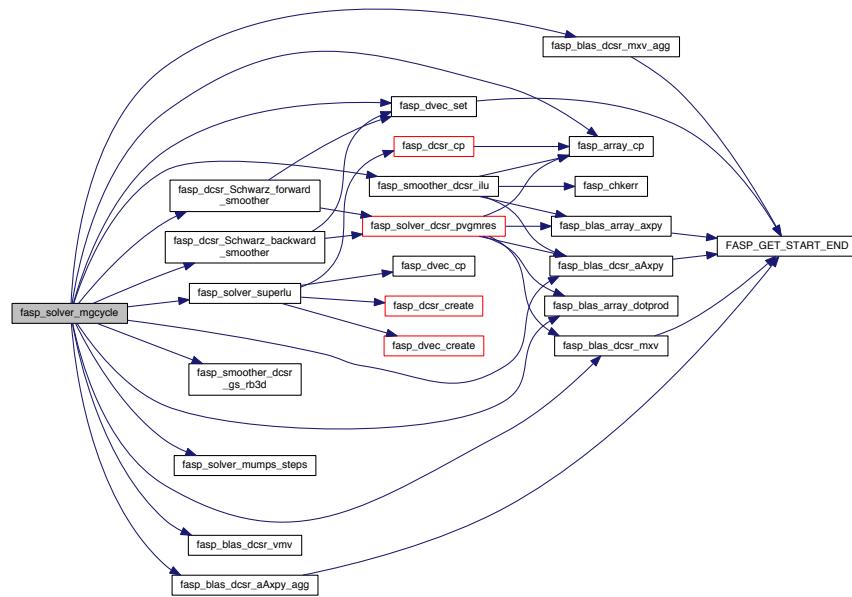
Date

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 41 of file mgcycle.c.

Here is the call graph for this function:



9.54.2.2 void fasp_solver_mgcycle_bsr (AMG_data_bsr * *mgl*, AMG_param * *param*)

Solve Ax=b with non-recursive multigrid cycle.

Parameters

<i>mgl</i>	Pointer to AMG data: AMG_data_bsr
<i>param</i>	Pointer to AMG parameters: AMG_param

Author

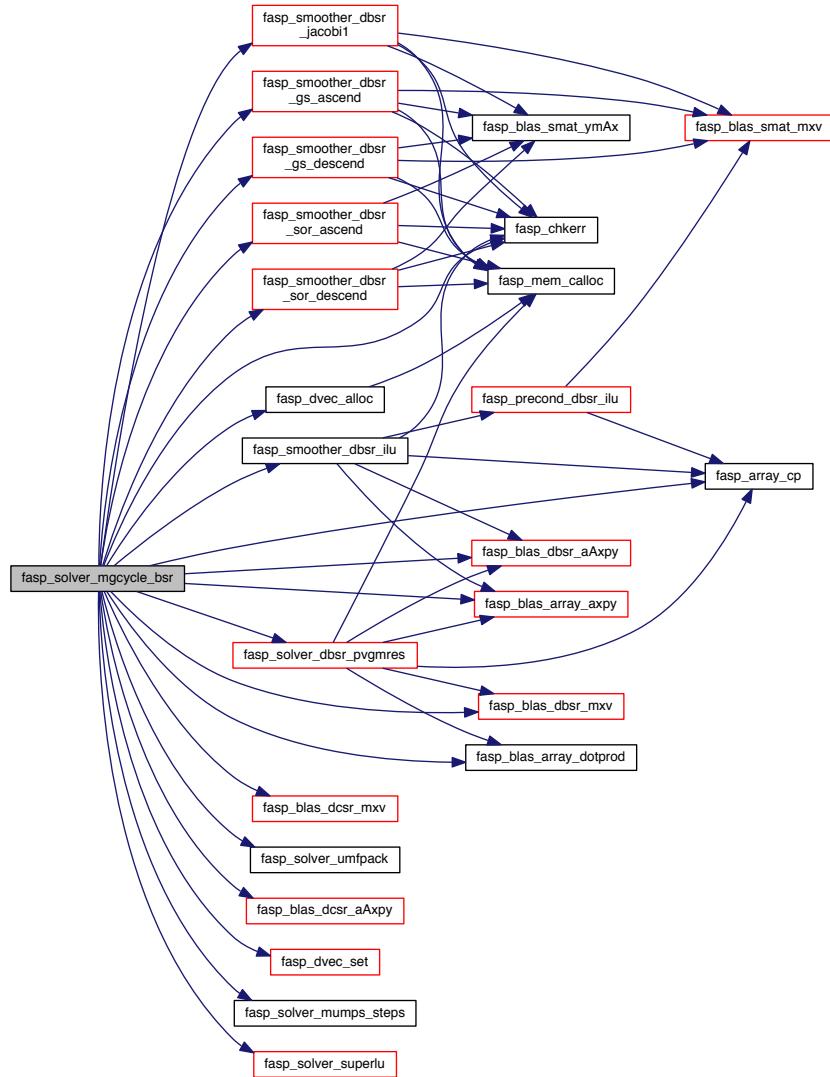
Xiaozhe Hu

Date

08/07/2011

Definition at line 257 of file mgcycle.c.

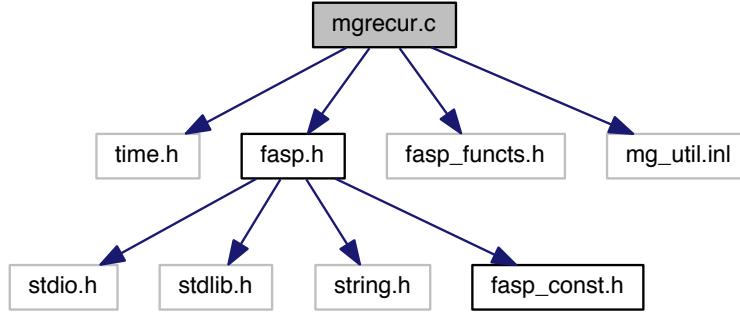
Here is the call graph for this function:



9.55 mgrecur.c File Reference

Abstract multigrid cycle – recursive version.

```
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "mg_util.inl"
Include dependency graph for mgrecur.c:
```



Functions

- void [fasp_solver_mgrecur \(AMG_data *mgl, AMG_param *param, INT level\)](#)
Solve Ax=b with recursive multigrid K-cycle.

9.55.1 Detailed Description

Abstract multigrid cycle – recursive version.

Note

Not used any more. Will be removed! –Chensong

9.55.2 Function Documentation

9.55.2.1 void [fasp_solver_mgrecur \(AMG_data * mgl, AMG_param * param, INT level \)](#)

Solve Ax=b with recursive multigrid K-cycle.

Parameters

<i>mgl</i>	Pointer to AMG data: AMG_data
<i>param</i>	Pointer to AMG parameters: AMG_param
<i>level</i>	Index of the current level

Author

Xuehai Huang, Chensong Zhang

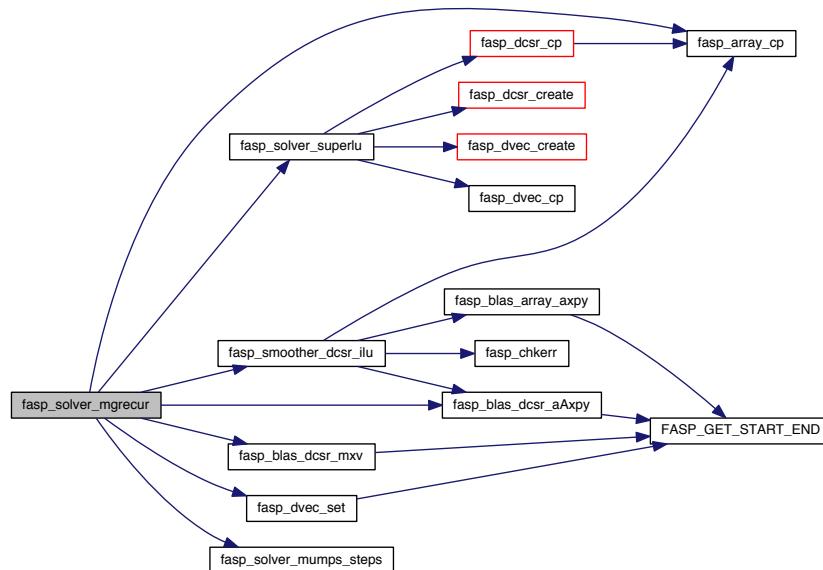
Date

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.

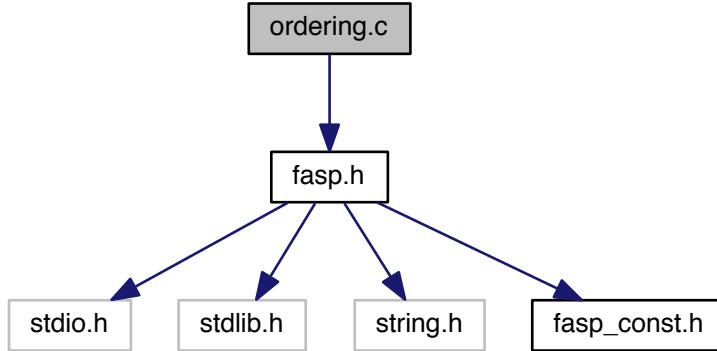
Here is the call graph for this function:



9.56 ordering.c File Reference

A collection of ordering, merging, removing duplicated integers functions.

```
#include "fasp.h"
Include dependency graph for ordering.c:
```



Functions

- `INT fasp_BinarySearch (INT *list, INT value, INT list_length)`
Binary Search.
- `INT fasp_aux_unique (INT numbers[], 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_msrot (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)`
Reverse CMK ordering.

9.56.1 Detailed Description

A collection of ordering, merging, removing duplicated integers functions.

9.56.2 Function Documentation

9.56.2.1 void fasp_aux_dQuickSort (REAL * a, INT left, INT right)

Sort the array (REAL type) in ascending order with the quick sorting algorithm.

Parameters

<i>a</i>	Pointer to the array needed to be sorted
<i>left</i>	Starting index
<i>right</i>	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 239 of file ordering.c.

9.56.2.2 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

<i>a</i>	Pointer to the array
<i>left</i>	Starting index
<i>right</i>	Ending index
<i>index</i>	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 320 of file ordering.c.

9.56.2.3 void fasp_aux_iQuickSort (INT * a, INT left, INT right)

Sort the array (INT type) in ascending order with the quick sorting algorithm.

Parameters

<i>a</i>	Pointer to the array needed to be sorted
<i>left</i>	Starting index
<i>right</i>	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 201 of file ordering.c.

9.56.2.4 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

<i>a</i>	Pointer to the array
<i>left</i>	Starting index
<i>right</i>	Ending index
<i>index</i>	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 279 of file ordering.c.

9.56.2.5 void fasp_aux_merge (INT *numbers*[], INT *work*[], INT *left*, INT *mid*, INT *right*)

Merge two sorted arrays.

Parameters

<i>numbers</i>	Pointer to the array needed to be sorted
<i>work</i>	Pointer to the work array with same size as numbers
<i>left</i>	Starting index of array 1
<i>mid</i>	Starting index of array 2
<i>right</i>	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 108 of file ordering.c.

9.56.2.6 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

<i>numbers</i>	Pointer to the array needed to be sorted
<i>work</i>	Pointer to the work array with same size as numbers
<i>left</i>	Starting index
<i>right</i>	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 170 of file ordering.c.

Here is the call graph for this function:



9.56.2.7 INT fasp_aux_unique (INT *numbers[]*, INT *size*)

Remove duplicates in an sorted (ascending order) array.

Parameters

<i>numbers</i>	Pointer to the array needed to be sorted (in/out)
<i>size</i>	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 75 of file ordering.c.

9.56.2.8 INT fasp_BinarySearch (INT * *list*, INT *value*, INT *list_length*)

Binary Search.

Parameters

<i>list</i>	Pointer to a set of values
<i>value</i>	The target
<i>list_length</i>	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 30 of file ordering.c.

9.56.2.9 void fasp_dcsr_CMK_order (const dCSRmat * *A*, INT * *order*, INT * *oindex*)

Ordering vertices of matrix graph corresponding to A.

Parameters

<i>A</i>	Pointer to matrix
<i>oindex</i>	Pointer to index of vertices in order
<i>order</i>	Pointer to vertices with increasing degree

Author

Zheng Li, Chensong Zhang

Date

05/28/2014

Definition at line 356 of file ordering.c.

9.56.2.10 void fasp_dcsr_RCMK_order (const dCSRmat * *A*, INT * *order*, INT * *oindex*, INT * *rorder*)

Resverse CMK ordering.

Parameters

<i>A</i>	Pointer to matrix
<i>order</i>	Pointer to vertices with increasing degree
<i>oindex</i>	Pointer to index of vertices in order
<i>rorder</i>	Pointer to reverse order

Author

Zheng Li, Chensong Zhang

Date

10/10/2014

Definition at line 405 of file ordering.c.

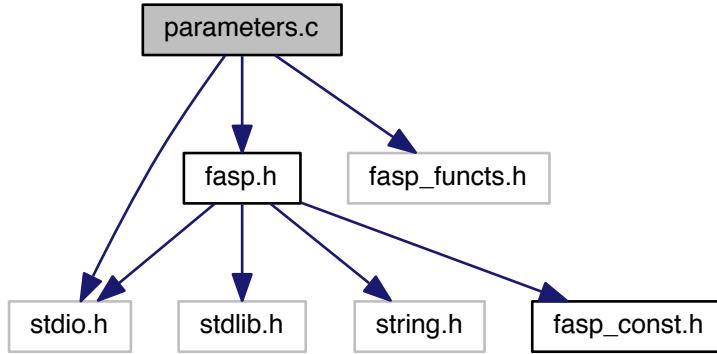
Here is the call graph for this function:



9.57 parameters.c File Reference

Initialize, set, or print input data and parameters.

```
#include <stdio.h>
#include "fasp.h"
#include "fasp_functs.h"
Include dependency graph for parameters.c:
```



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.

9.57.1 Detailed Description

Initialize, set, or print input data and parameters.

9.57.2 Function Documentation

9.57.2.1 void `fasp_param_amg_init` (`AMG_param` * `amgparam`)

Initialize AMG parameters.

Parameters

<code>amgparam</code>	Parameters for AMG
-----------------------	--------------------

Author

Chensong Zhang

Date

2010/04/03

Definition at line 389 of file `parameters.c`.

9.57.2.2 void `fasp_param_amg_print` (`AMG_param` * `param`)

Print out AMG parameters.

Parameters

<i>param</i>	Parameters for AMG
--------------	--------------------

Author

Chensong Zhang

Date

2010/03/22

Definition at line 794 of file parameters.c.

9.57.2.3 void fasp_param_amg_set (AMG_param * param, input_param * iniparam)

Set [AMG_param](#) from INPUT.

Parameters

<i>param</i>	Parameters for AMG
<i>iniparam</i>	Input parameters

Author

Chensong Zhang

Date

2010/03/23

Definition at line 516 of file parameters.c.

9.57.2.4 void fasp_param_amg_to_prec (precond_data * pcdata, AMG_param * amgparam)

Set [precond_data](#) with [AMG_param](#).

Parameters

<i>pcdata</i>	Preconditioning data structure
<i>amgparam</i>	Parameters for AMG

Author

Chensong Zhang

Date

2011/01/10

Definition at line 663 of file parameters.c.

9.57.2.5 void fasp_param_amg_to_prec_bsr (precond_data_bsr * pcdata, AMG_param * amgparam)

Set [precond_data_bsr](#) with [AMG_param](#).

Parameters

<i>pcdata</i>	Preconditioning data structure
<i>amgparam</i>	Parameters for AMG

Author

Xiaozhe Hu

Date

02/06/2012

Definition at line 730 of file parameters.c.

9.57.2.6 void fasp_param_ilu_init (ILU_param * *iluparam*)

Initialize ILU parameters.

Parameters

<i>iluparam</i>	Parameters for ILU
-----------------	--------------------

Author

Chensong Zhang

Date

2010/04/06

Definition at line 474 of file parameters.c.

9.57.2.7 void fasp_param_ilu_print (ILU_param * *param*)

Print out ILU parameters.

Parameters

<i>param</i>	Parameters for ILU
--------------	--------------------

Author

Chensong Zhang

Date

2011/12/20

Definition at line 894 of file parameters.c.

9.57.2.8 void fasp_param_ilu_set (ILU_param * *iluparam*, input_param * *iniparam*)

Set [ILU_param](#) with INPUT.

Parameters

<i>iluparam</i>	Parameters for ILU
<i>iniparam</i>	Input parameters

Author

Chensong Zhang

Date

2010/04/03

Definition at line 590 of file parameters.c.

9.57.2.9 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

<i>iniparam</i>	Input parameters
<i>itsparam</i>	Iterative solver parameters
<i>amgparam</i>	AMG parameters
<i>iluparam</i>	ILU parameters
<i>schparam</i>	Schwarz parameters

Author

Chensong Zhang

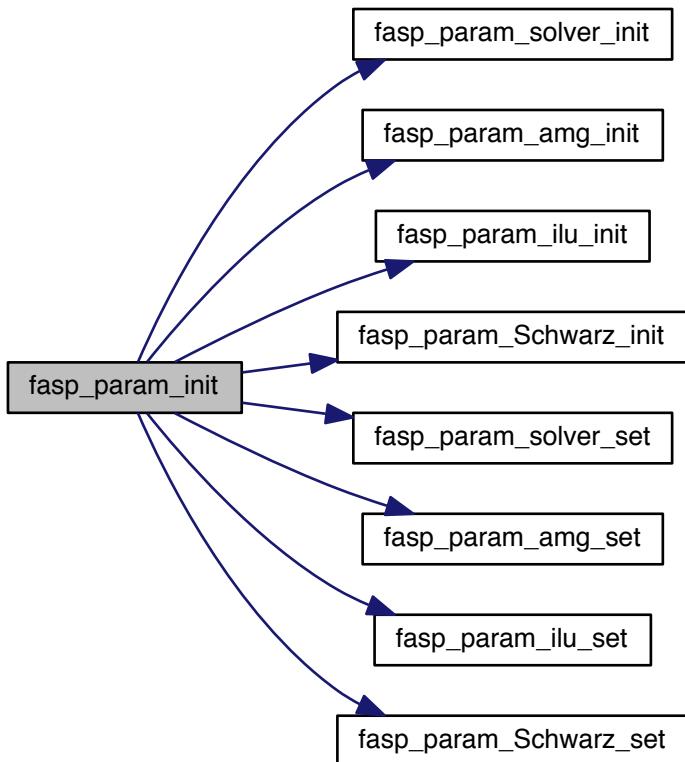
Date

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.

Here is the call graph for this function:



9.57.2.10 void fasp_param_input_init (*input_param* * *iniparam*)

Initialize input parameters.

Parameters

<i>iniparam</i>	Input parameters
-----------------	------------------

Author

Chensong Zhang

Date

2010/03/20

Definition at line 310 of file parameters.c.

9.57.2.11 void fasp_param_prec_to_amg (**AMG_param** * *amgparam*, **precond_data** * *pcdata*)

Set **AMG_param** with **precond_data**.

Parameters

<i>amgparam</i>	Parameters for AMG
<i>pcdata</i>	Preconditioning data structure

Author

Chensong Zhang

Date

2011/01/10

Definition at line 698 of file parameters.c.

9.57.2.12 void fasp_param_prec_to_amg_bsr (**AMG_param** * *amgparam*, **precond_data_bsr** * *pcdata*)

Set **AMG_param** with **precond_data**.

Parameters

<i>amgparam</i>	Parameters for AMG
<i>pcdata</i>	Preconditioning data structure

Author

Xiaozhe Hu

Date

02/06/2012

Definition at line 764 of file parameters.c.

9.57.2.13 void fasp_param_Schwarz_init (**Schwarz_param** * *schparam*)

Initialize Schwarz parameters.

Parameters

<i>schparam</i>	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 496 of file parameters.c.

9.57.2.14 void fasp_param_Schwarz_print (Schwarz_param * *param*)

Print out Schwarz parameters.

Parameters

<i>param</i>	Parameters for Schwarz
--------------	------------------------

Author

Xiaozhe Hu

Date

05/22/2012

Definition at line 924 of file parameters.c.

9.57.2.15 void fasp_param_Schwarz_set (Schwarz_param * *schparam*, input_param * *iniparam*)

Set [Schwarz_param](#) with INPUT.

Parameters

<i>schparam</i>	Parameters for Schwarz method
<i>iniparam</i>	Input parameters

Author

Xiaozhe Hu

Date

05/22/2012

Definition at line 612 of file parameters.c.

9.57.2.16 void fasp_param_set (int *argc*, const char * *argv*[], input_param * *iniparam*)

Read input from command-line arguments.

Parameters

<i>argc</i>	Number of arg input
<i>argv</i>	Input arguments
<i>iniparam</i>	Parameters to be set

Author

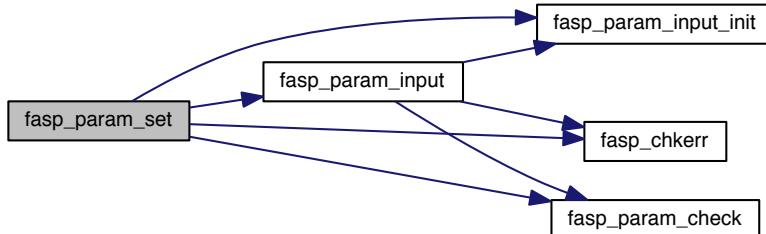
Chensong Zhang

Date

12/29/2013

Definition at line 27 of file parameters.c.

Here is the call graph for this function:



9.57.2.17 void fasp_param_solver_init (*itsolver_param* * *itsparam*)

Initialize *itsolver_param*.

Parameters

<i>itsparam</i>	Parameters for iterative solvers
-----------------	----------------------------------

Author

Chensong Zhang

Date

2010/03/23

Definition at line 453 of file parameters.c.

9.57.2.18 void fasp_param_solver_print (*itsolver_param* * *param*)

Print out itsolver parameters.

Parameters

<i>param</i>	Paramters for iterative solvers
--------------	---------------------------------

Author

Chensong Zhang

Date

2011/12/20

Definition at line 953 of file parameters.c.

9.57.2.19 void fasp_param_solver_set(*itsolver_param* * *itsparam*, *input_param* * *iniparam*)

Set *itsolver_param* with INPUT.

Parameters

<i>itsparam</i>	Parameters for iterative solvers
<i>iniparam</i>	Input parameters

Author

Chensong Zhang

Date

2010/03/23

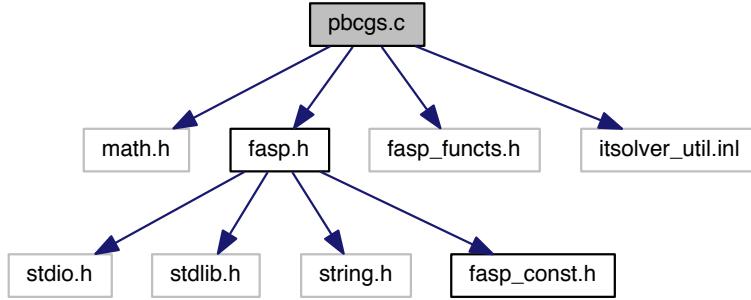
Definition at line 633 of file parameters.c.

9.58 pbcgs.c File Reference

Krylov subspace methods – Preconditioned BiCGstab.

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

Include dependency graph for pbcgs.c:



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 print_level)`
Preconditioned BiCGstab method for solving Au=b.
- `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 print_level)`
Preconditioned BiCGstab method for solving Au=b.
- `INT fasp_solver_bdcsr_pbcgs (block_dCSRmat *A, dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT print_level)`
A preconditioned BiCGstab method for solving Au=b.
- `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 print_level)`
Preconditioned BiCGstab method for solving Au=b.

9.58.1 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 $V_k = \text{span}\{p_1, \dots, p_k\}$.

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 $k = 0:\text{MaxIt}$

- 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 $\{p_0, p_1, \dots, p_k\}$;
- prepare for next iteration;
- print the result of k-th iteration; END FOR

Convergence check: $\text{norm}(r)/\text{norm}(b) < \text{tol}$

Stagnation check:

- IF $\text{norm}(\alpha * p_k)/\text{norm}(x_{k+1}) < \text{tol_stag}$
 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 $\text{norm}(r_{k+1})/\text{norm}(b) < \text{tol}$
 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

9.58.2 Function Documentation

9.58.2.1 INT fasp_solver_bdcsl_pbcgs (block_dCSRmat * A, dvector * b, dvector * u, precond * pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT print_level)

A preconditioned BiCGstab method for solving $Au=b$.

Parameters

<i>A</i>	Pointer to the coefficient matrix
<i>b</i>	Pointer to the dvector of right hand side

<i>u</i>	Pointer to the dvector of DOFs
<i>pc</i>	Pointer to precond: the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Xiaozhe Hu

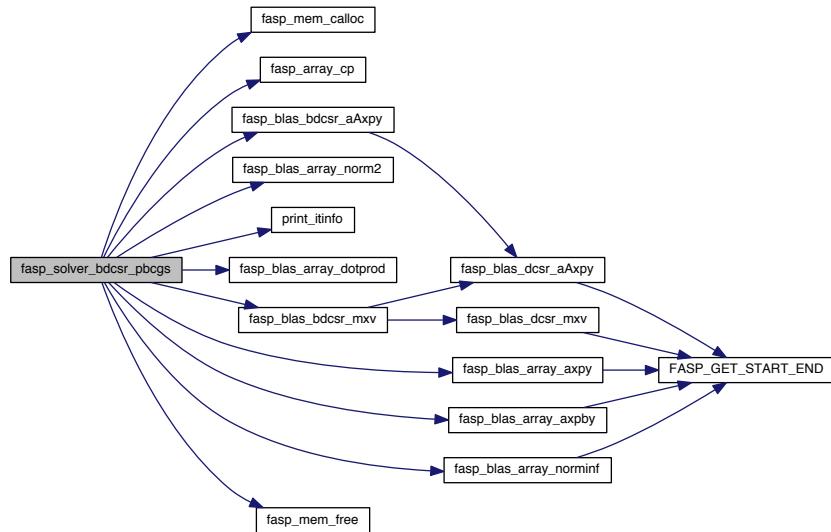
Date

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 774 of file pbcgs.c.

Here is the call graph for this function:



9.58.2.2 INT fasp_solver_dbsr_pbcgs (dBsrMat * A, dvector * b, dvector * u, precondition * pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT print_level)

Preconditioned BiCGstab method for solving Au=b.

Parameters

<i>A</i>	Pointer to the coefficient matrix
<i>b</i>	Pointer to the dvector of right hand side
<i>u</i>	Pointer to the dvector of DOFs
<i>pc</i>	Pointer to precond: the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Chensong Zhang

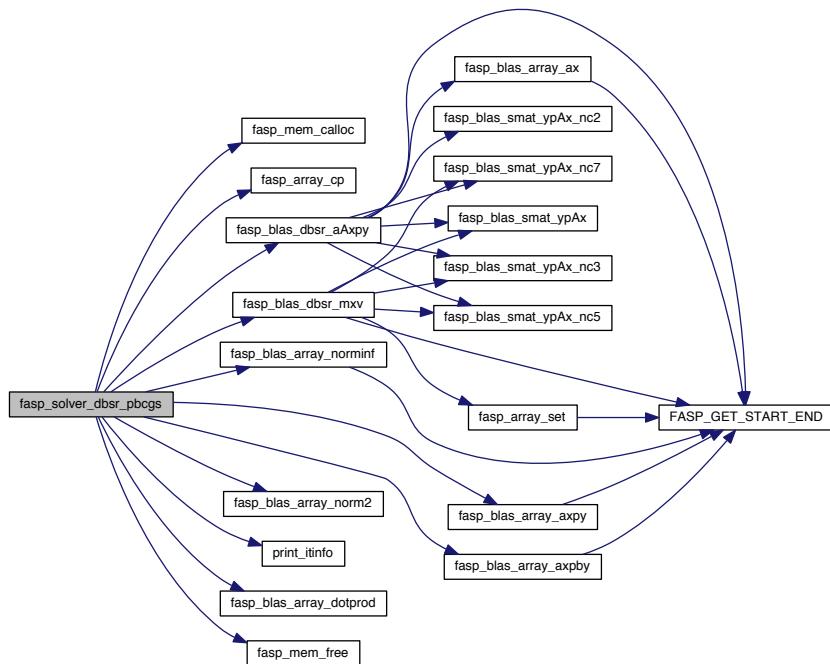
Date

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 431 of file pbcgs.c.

Here is the call graph for this function:



9.58.2.3 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 *print_level*)

Preconditioned BiCGstab method for solving $Au=b$.

Parameters

<i>A</i>	Pointer to the coefficient matrix
<i>b</i>	Pointer to the dvector of right hand side
<i>u</i>	Pointer to the dvector of DOFs
<i>pc</i>	Pointer to precond: the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Chensong Zhang

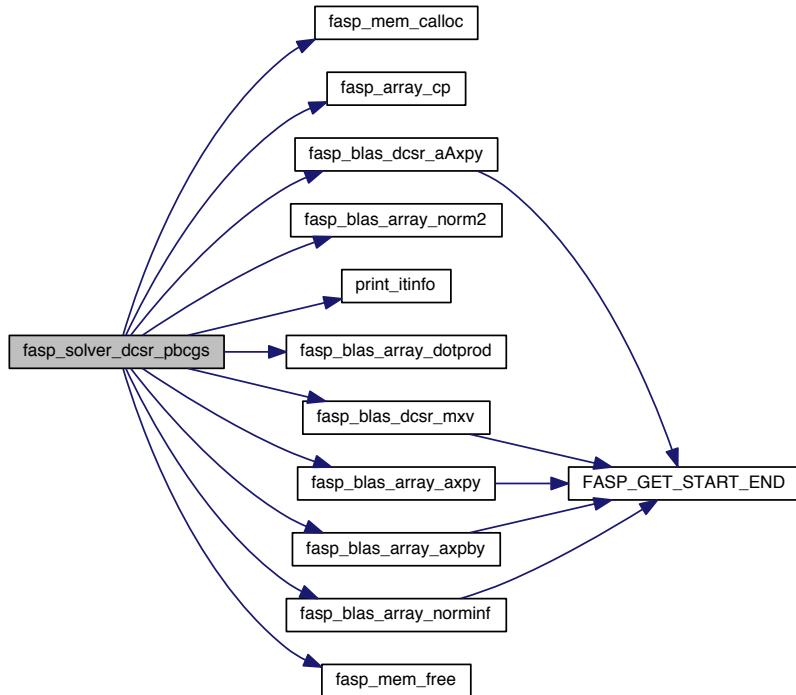
Date

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 88 of file pbcgs.c.

Here is the call graph for this function:



9.58.2.4 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 *print_level*)

Preconditioned BiCGstab method for solving $Au=b$.

Parameters

<i>A</i>	Pointer to the coefficient matrix
<i>b</i>	Pointer to the dvector of right hand side
<i>u</i>	Pointer to the dvector of DOFs
<i>pc</i>	Pointer to precond: the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Zhiyang Zhou

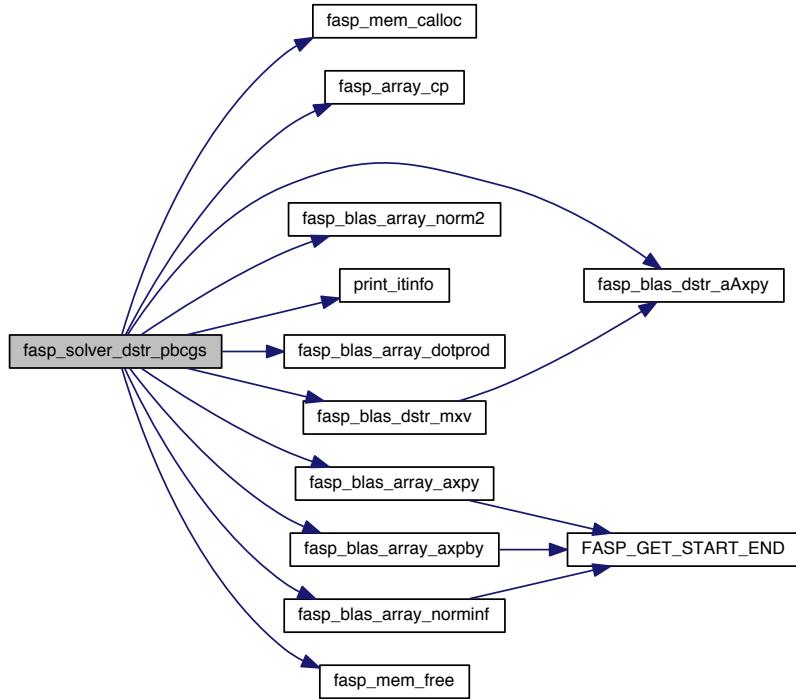
Date

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 1117 of file pbcgs.c.

Here is the call graph for this function:

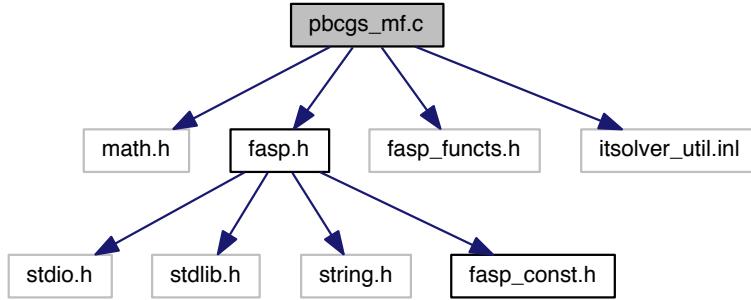


9.59 pbcgs_mf.c File Reference

Krylov subspace methods – Preconditioned BiCGstab (matrix free)

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

Include dependency graph for pbcgs_mf.c:



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 print_level)**

Preconditioned BiCGstab method for solving $Au=b$.

9.59.1 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

$V_k = \text{span}\{r_0, A*r_0, A^2*r_0, \dots, A^{k-1}*r_0\}$,

under some inner product.

For the implementation, we generate a series of $\{p_k\}$ such that $V_k = \text{span}\{p_1, \dots, p_k\}$. 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 $k = 0:\text{MaxIt}$

- 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 $\{p_0, p_1, \dots, p_k\}$;
- prepare for next iteration;
- print the result of k-th iteration; END FOR

Convergence check is: $\text{norm}(r)/\text{norm}(b) < \text{tol}$

Stagnation check is like following:

- IF $\text{norm}(\alpha * p_k)/\text{norm}(x_{\{k+1\}}) < \text{tol_stag}$
 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 $\text{norm}(r_{\{k+1\}})/\text{norm}(b) < \text{tol}$
 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

9.59.2 Function Documentation

9.59.2.1 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 *print_level*)

Preconditioned BiCGstab method for solving $Au=b$.

Parameters

<i>mf</i>	Pointer to mxv_matfree : the spmv operation
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to precond: the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Chensong Zhang

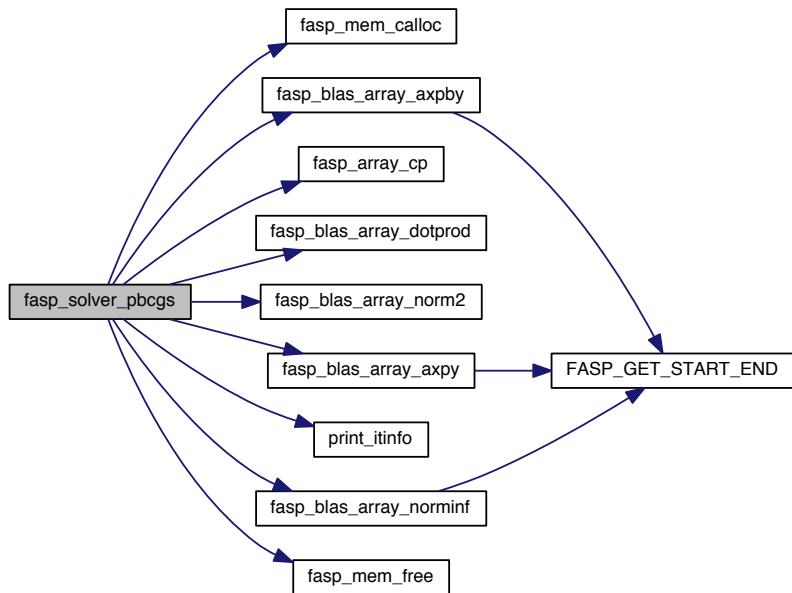
Date

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 91 of file pbcgs_mf.c.

Here is the call graph for this function:

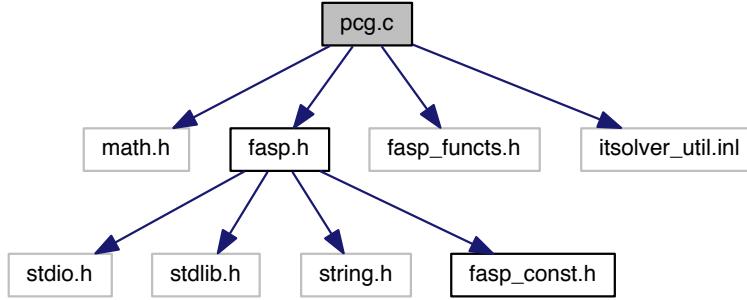


9.60 pcg.c File Reference

Krylov subspace methods – Preconditioned conjugate gradient.

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

Include dependency graph for pcg.c:



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 print_level)`
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 print_level)`
Preconditioned conjugate gradient method for solving $Au=b$.
- `INT fasp_solver_bdcsr_pcg (block_dCSRmat *A, dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT print_level)`
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 print_level)`
Preconditioned conjugate gradient method for solving $Au=b$.

9.60.1 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 $\{p_0, p_1, \dots, p_k\}$;
- prepare for next iteration;
- print the result of k-th iteration; END FOR

Convergence check: $\text{norm}(r)/\text{norm}(b) < \text{tol}$

Stagnation check:

- IF $\text{norm}(\alpha * p_k)/\text{norm}(x_{\{k+1\}}) < \text{tol_stag}$
 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 $\text{norm}(r_{\{k+1\}})/\text{norm}(b) < \text{tol}$
 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

9.60.2 Function Documentation

9.60.2.1 INT fasp_solver_bdcsr_pcg (**block_dCSRmat** * *A*, **dvector** * *b*, **dvector** * *u*, **precond** * *pc*, const **REAL** *tol*, const **INT** *MaxIt*, const **SHORT** *stop_type*, const **SHORT** *print_level*)

Preconditioned conjugate gradient method for solving $Au=b$.

Parameters

<i>A</i>	Pointer to block_dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector : the right hand side
<i>u</i>	Pointer to dvector : the unknowns

<i>pc</i>	Pointer to precond: the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Xiaozhe Hu

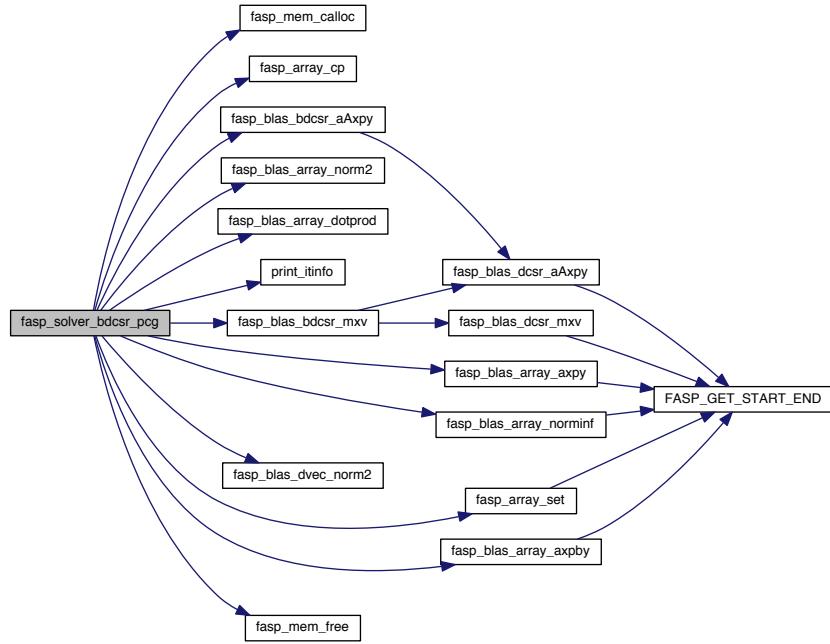
Date

05/24/2010

Modified by Chensong Zhang on 04/30/2012 Modified by Chensong Zhang on 03/28/2013

Definition at line 651 of file pcg.c.

Here is the call graph for this function:



9.60.2.2 INT fasp_solver_dbsr_pcg (dBsrmat * A, dvector * b, dvector * u, precondition * pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT print_level)

Preconditioned conjugate gradient method for solving Au=b.

Parameters

<i>A</i>	Pointer to dBSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to precond: the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

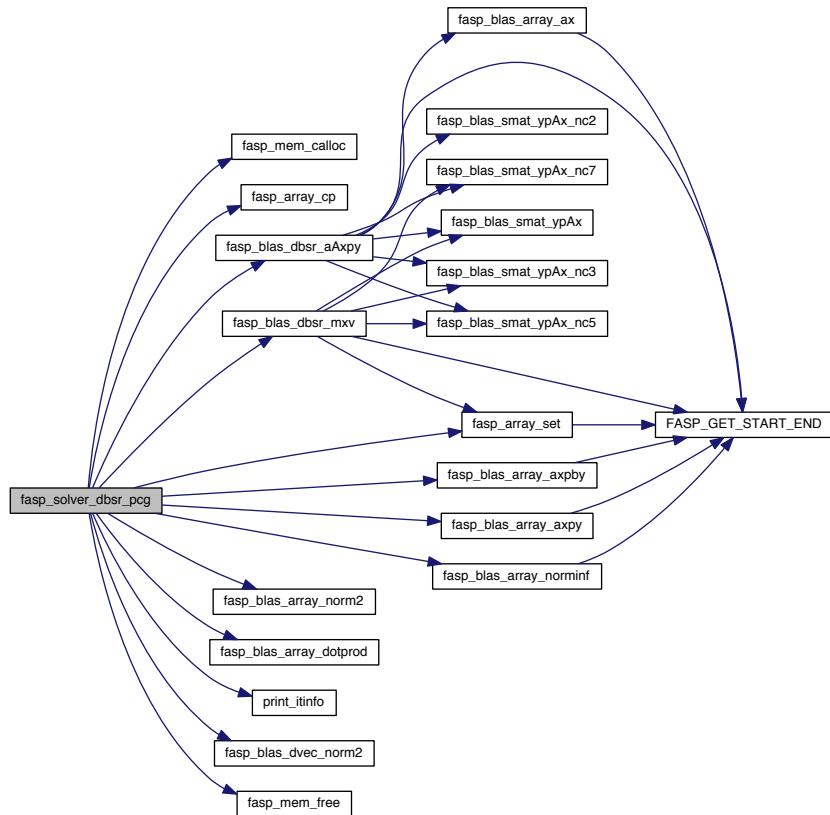
Xiaozhe Hu

Date

05/26/2014

Definition at line 367 of file pcg.c.

Here is the call graph for this function:



9.60.2.3 **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 *print_level*)**

Preconditioned conjugate gradient method for solving $Au=b$.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to precond: the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Chensong Zhang, Xiaozhe Hu, Shiquan Zhang

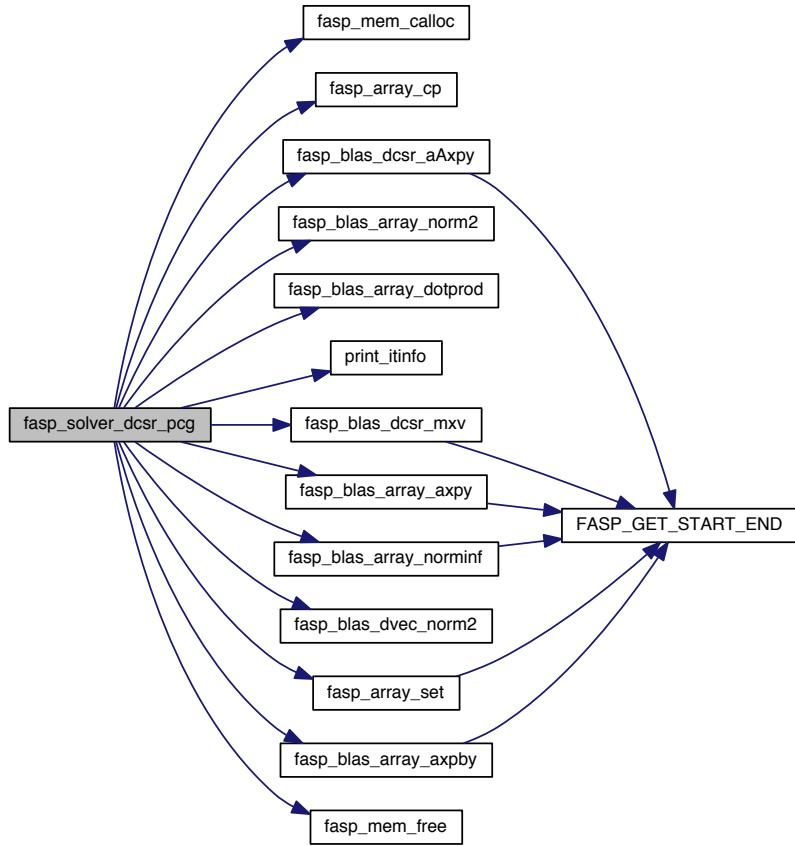
Date

05/06/2010

Modified by Chensong Zhang on 04/30/2012 Modified by Chensong Zhang on 03/28/2013

Definition at line 85 of file pcg.c.

Here is the call graph for this function:



9.60.2.4 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 print_level)

Preconditioned conjugate gradient method for solving $Au=b$.

Parameters

<i>A</i>	Pointer to dSTRmat : the coefficient matrix
<i>b</i>	Pointer to dvector : the right hand side
<i>u</i>	Pointer to dvector : the unknowns
<i>pc</i>	Pointer to precond : the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations

<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Zhiyang Zhou

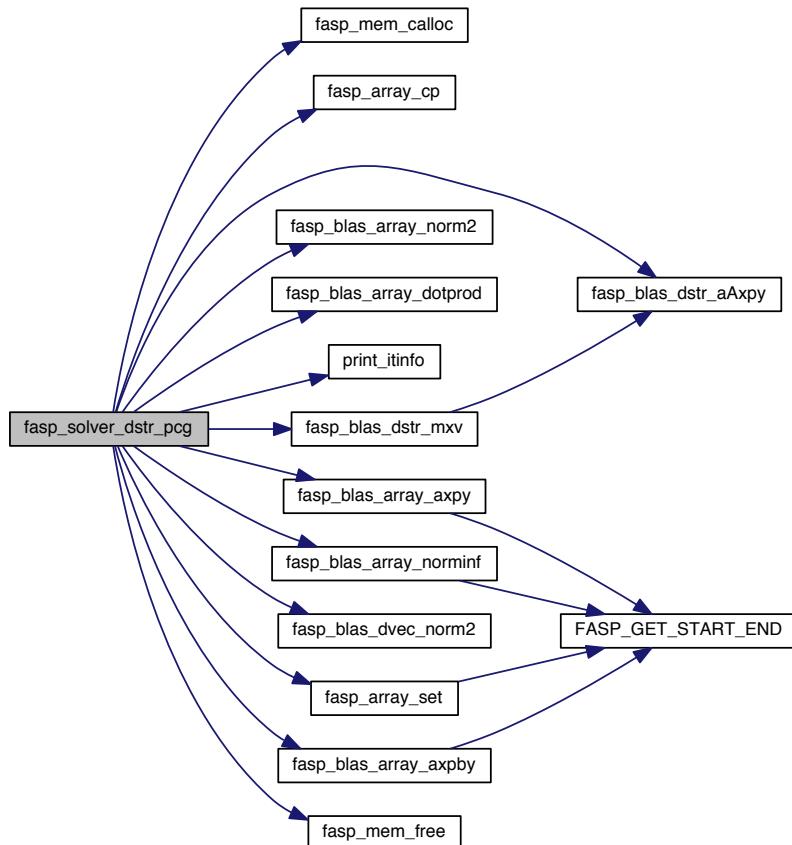
Date

04/25/2010

Modified by Chensong Zhang on 04/30/2012 Modified by Chensong Zhang on 03/28/2013

Definition at line 935 of file pcg.c.

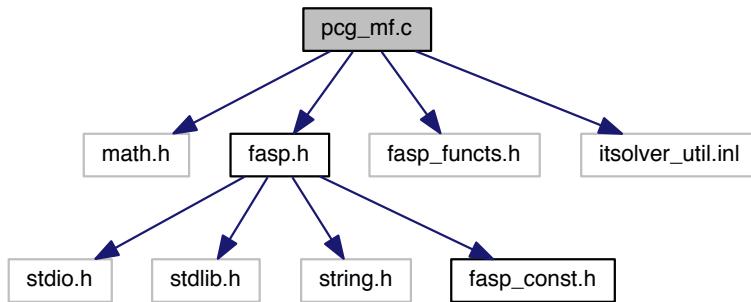
Here is the call graph for this function:



9.61 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"
Include dependency graph for pcg_mf.c:
```



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 print_level)**

Preconditioned conjugate gradient (CG) method for solving Au=b.

9.61.1 Detailed Description

Krylov subspace methods – Preconditioned conjugate gradient (matrix free)

Abstract algorithm

PCG method to solve $A \cdot x = b$ is to generate $\{x_k\}$ to approximate x

Step 0. Given A , b , x_0 , M

Step 1. Compute residual $r_0 = b - A \cdot x_0$ and convergence check;

Step 2. Initialization $z_0 = M^{-1} \cdot r_0$, $p_0 = z_0$;

Step 3. Main loop ...

FOR $k = 0$: $MaxIt$

- get step size $\alpha = f(r_k, z_k, p_k)$;
- update solution: $x_{k+1} = x_k + \alpha \cdot p_k$;
- perform stagnation check;

- update residual: $r_{\{k+1\}} = r_k - \alpha * (A * p_k)$;
- perform residual check;
- obtain $p_{\{k+1\}}$ using $\{p_0, p_1, \dots, p_k\}$;
- prepare for next iteration;
- print the result of k-th iteration; END FOR

Convergence check is: $\text{norm}(r)/\text{norm}(b) < \text{tol}$

Stagnation check is like following:

- IF $\text{norm}(\alpha * p_k)/\text{norm}(x_{\{k+1\}}) < \text{tol_stag}$
 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 $\text{norm}(r_{\{k+1\}})/\text{norm}(b) < \text{tol}$
 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

9.61.2 Function Documentation

9.61.2.1 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 *print_level*)

Preconditioned conjugate gradient (CG) method for solving $Au=b$.

Parameters

<i>mf</i>	Pointer to mxv_matfree : the spmv operation
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to precond: the structure of precondition
<i>tol</i>	Tolerance for stopping

<i>MaxIt</i>	Maximal number of iterations
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Chensong Zhang, Xiaozhe Hu, Shiquan Zhang

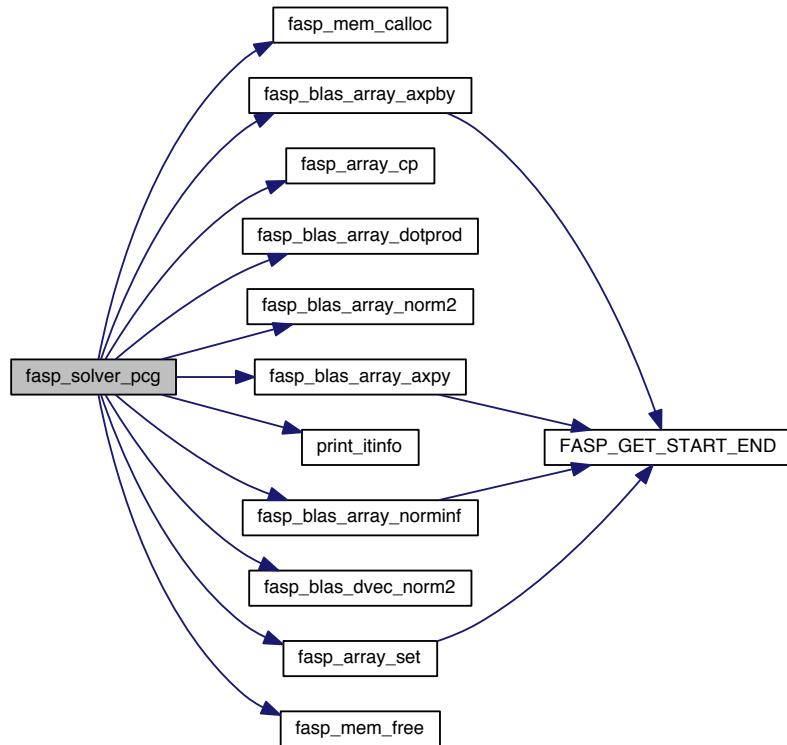
Date

05/06/2010

Modified by Chensong Zhang on 04/30/2012 Modified by Feiteng Huang on 09/19/2012: matrix free

Definition at line 87 of file pcg_mf.c.

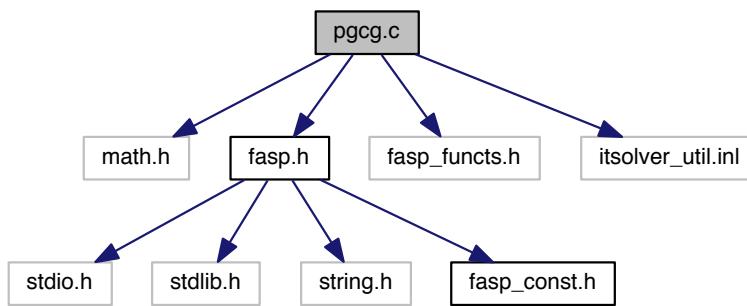
Here is the call graph for this function:



9.62 pgcg.c File Reference

Krylov subspace methods – Preconditioned Generalized CG.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
Include dependency graph for pgcg.c:
```



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 print_level)`

Preconditioned generalized conjugate gradient (GCG) method for solving $Au=b$.

9.62.1 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

9.62.2 Function Documentation

- 9.62.2.1 `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 print_level)`

Preconditioned generalized conjugate gradient (GCG) method for solving $Au=b$.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to precond: the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Xiaozhe Hu

Date

01/01/2012

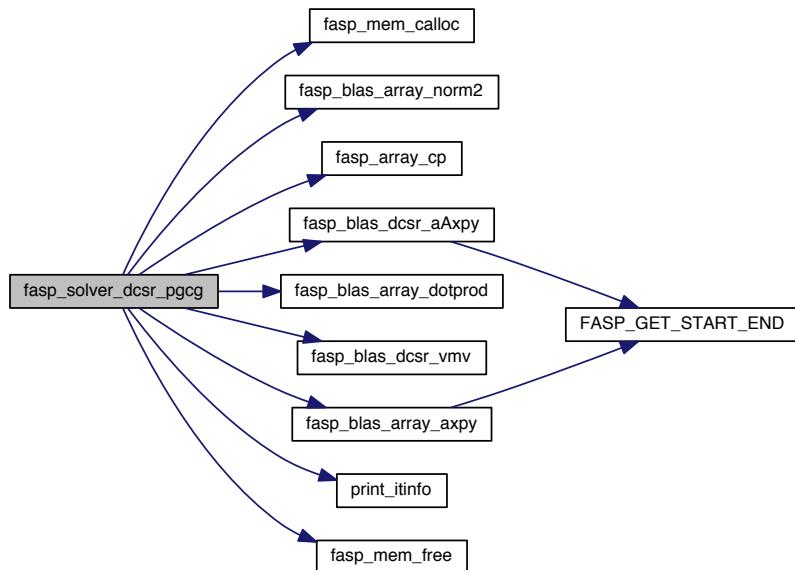
Note

Not completely implemented yet! –Chensong

Modified by Chensong Zhang on 05/01/2012

Definition at line 46 of file pgcg.c.

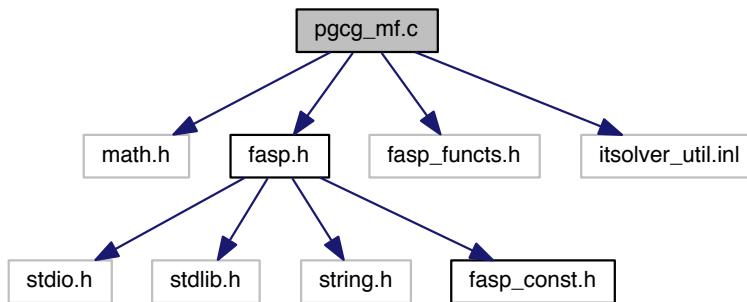
Here is the call graph for this function:



9.63 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"
Include dependency graph for pgcg_mf.c:
```



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 print_level)`

Preconditioned generalized conjugate gradient (GCG) method for solving $Au=b$.

9.63.1 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

9.63.2 Function Documentation

9.63.2.1 `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 print_level)`

Preconditioned generalized conjugate gradient (GCG) method for solving $Au=b$.

Parameters

<i>mf</i>	Pointer to mxv_matfree : the spmv operation
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to precond: the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>stop_type</i>	Stopping criteria type – Not implemented
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Xiaozhe Hu

Date

01/01/2012

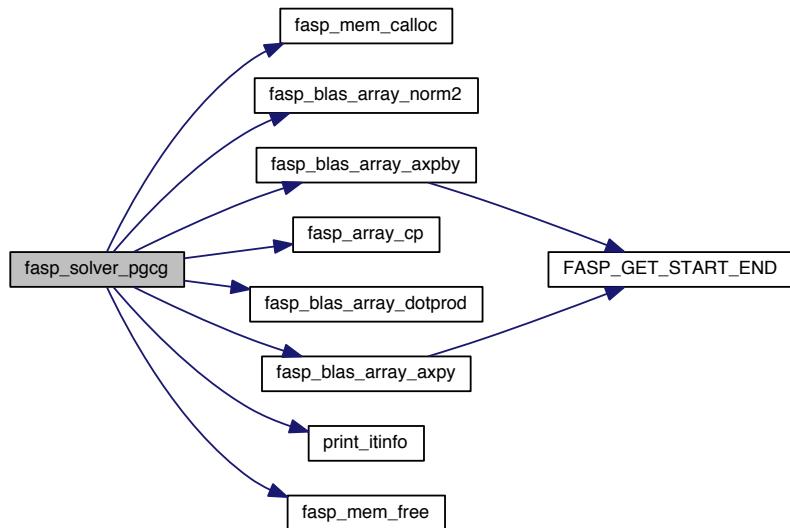
Note

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.

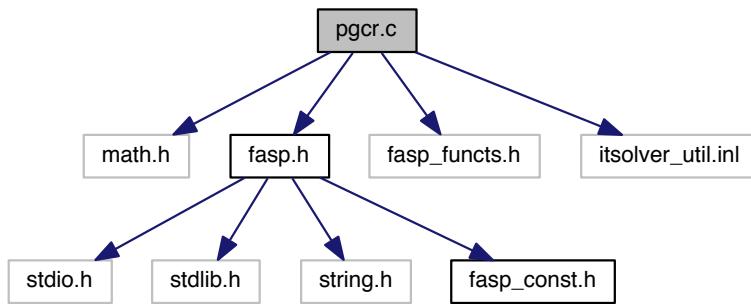
Here is the call graph for this function:



9.64 pgcr.c File Reference

Krylov subspace methods – Preconditioned GCR.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
Include dependency graph for pgcr.c:
```



Functions

- `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 $Au=b$.
- `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 $Au=b$.

9.64.1 Detailed Description

Krylov subspace methods – Preconditioned GCR.

9.64.2 Function Documentation

9.64.2.1 `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

<i>A</i>	Pointer to the coefficient matrix
<i>b</i>	Pointer to the dvector of right hand side
<i>x</i>	Pointer to the dvector of dofs
<i>pc</i>	Pointer to the structure of precondition (precond)
<i>tol</i>	Tolerance for stoppage
<i>MaxIt</i>	Maximal number of iterations
<i>restart</i>	Restart number for GCR
<i>stop_type</i>	Stopping type
<i>prtlvl</i>	How much information to print out

Returns

the number of iterations

Author

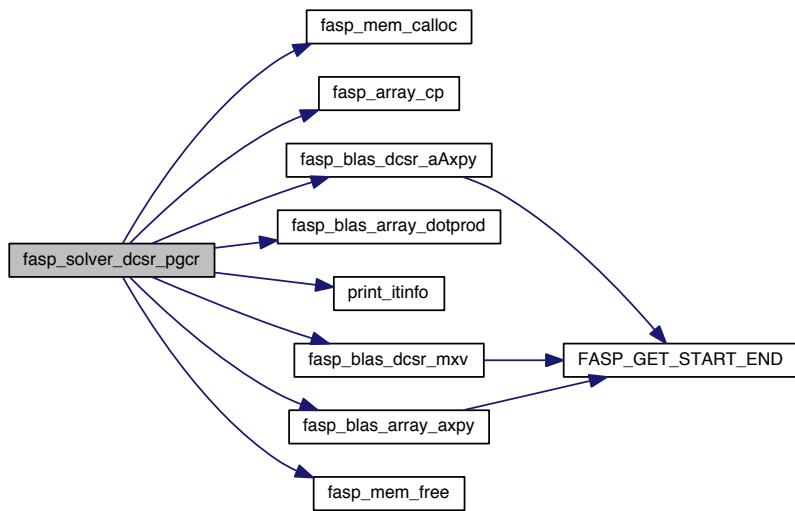
Zheng Li

Date

12/23/2014

Definition at line 248 of file pgcr.c.

Here is the call graph for this function:



9.64.2.2 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

<i>A</i>	Pointer to the coefficient matrix
<i>b</i>	Pointer to the dvector of right hand side
<i>x</i>	Pointer to the dvector of dofs
<i>pc</i>	Pointer to the structure of precondition (precond)
<i>tol</i>	Tolerance for stopage
<i>MaxIt</i>	Maximal number of iterations
<i>restart</i>	Restart number for GCR
<i>stop_type</i>	Stopping type
<i>prtlvl</i>	How much information to print out

Returns

the number of iterations

Author

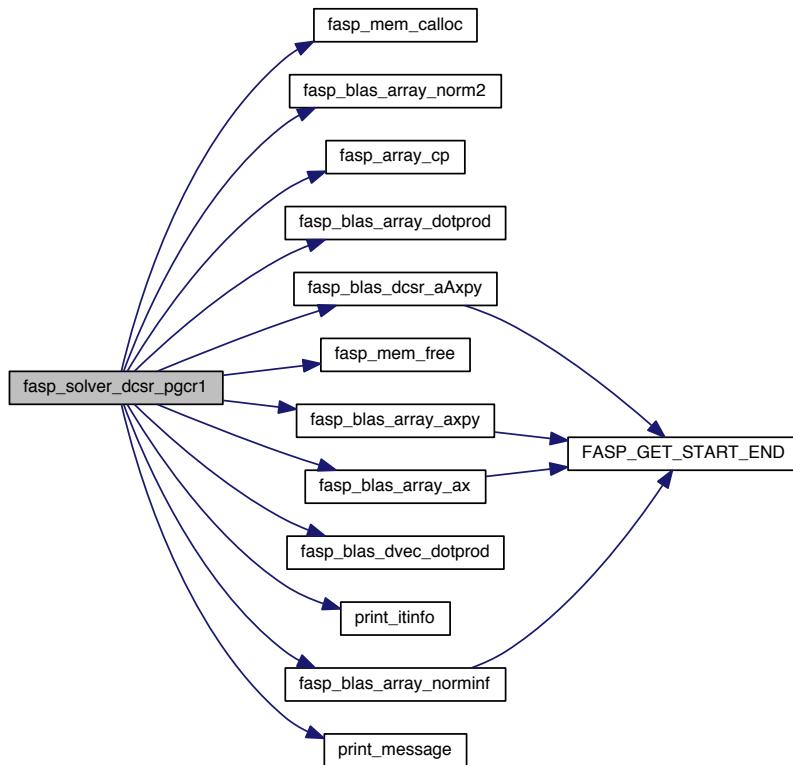
Lu Wang

Date

11/02/2014

Definition at line 35 of file pgcr.c.

Here is the call graph for this function:

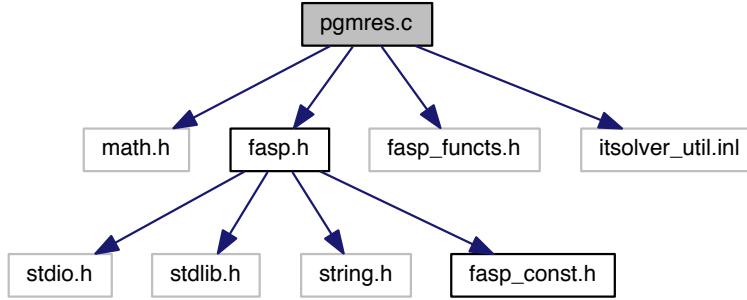


9.65 pgmres.c File Reference

Krylov subspace methods – Right-preconditioned GMRes.

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

Include dependency graph for pgmres.c:



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 print_level)`
Right preconditioned GMRES method for solving $Au=b$.
- `INT fasp_solver_bdcsr_pgmres (block_dCSRmat *A, dvector *b, dvector **x, precond *pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT print_level)`
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 print_level)`
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 print_level)`
Preconditioned GMRES method for solving $Au=b$.

9.65.1 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

9.65.2 Function Documentation

9.65.2.1 `INT fasp_solver_bdcsr_pgmres (block_dCSRmat * A, dvector * b, dvector ** x, precond * pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT print_level)`

Preconditioned GMRES method for solving $Au=b$.

Parameters

<i>A</i>	Pointer to <code>block_dCSRmat</code> : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>x</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to precond: the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>restart</i>	Restarting steps
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Xiaozhe Hu

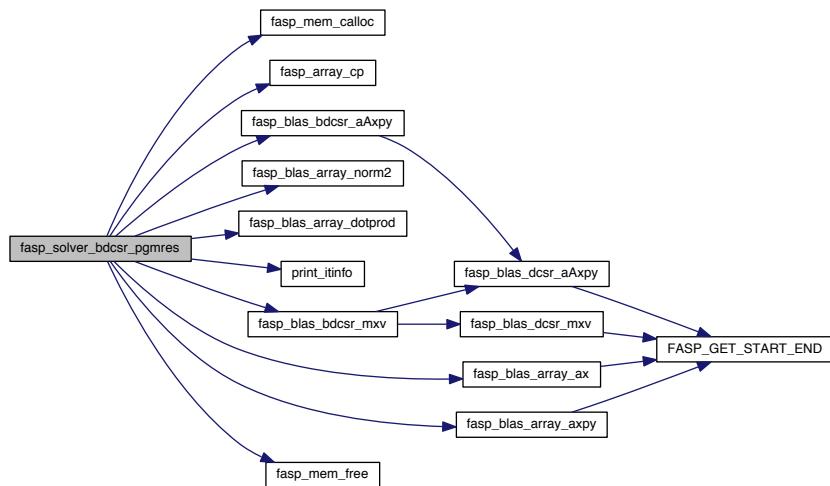
Date

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.

Here is the call graph for this function:



9.65.2.2 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 print_level*)

Preconditioned GMRES method for solving $Au=b$.

Parameters

<i>A</i>	Pointer to dBSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>x</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to precond: the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>restart</i>	Restarting steps
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Zhiyang Zhou

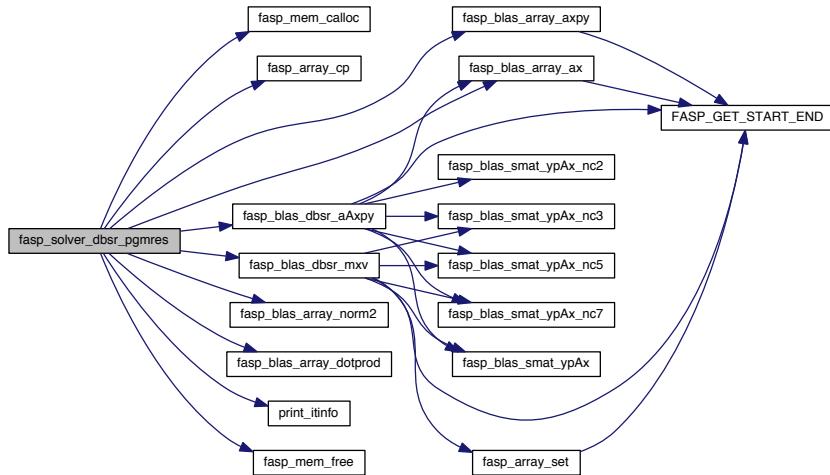
Date

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.

Here is the call graph for this function:



9.65.2.3 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 *print_level*)

Right preconditioned GMRES method for solving $Au=b$.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>x</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to precond: the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>restart</i>	Restarting steps
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Zhiyang Zhou

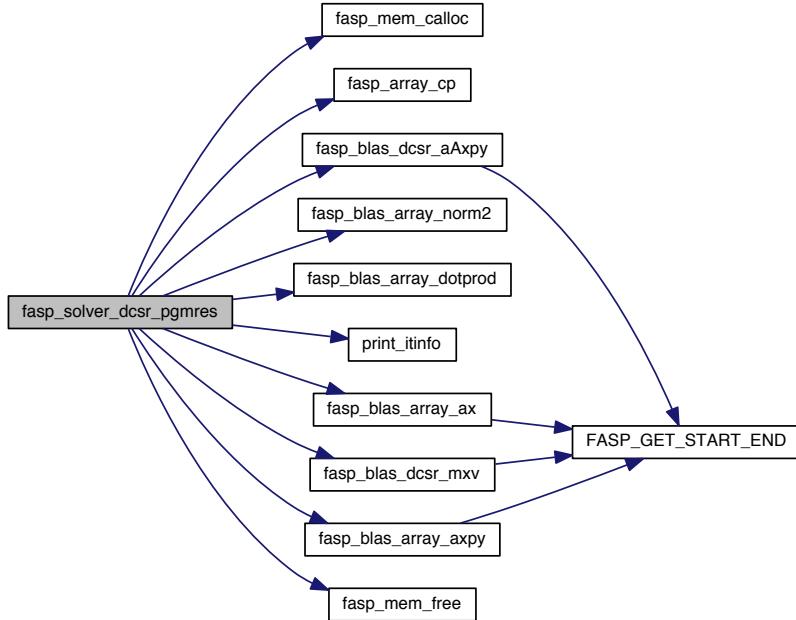
Date

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.

Here is the call graph for this function:



9.65.2.4 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 *print_level*)

Preconditioned GMRES method for solving $Au=b$.

Parameters

<i>A</i>	Pointer to dSTRmat : the coefficient matrix
<i>b</i>	Pointer to dvector : the right hand side
<i>x</i>	Pointer to dvector : the unknowns
<i>pc</i>	Pointer to precond : the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>restart</i>	Restarting steps
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Zhiyang Zhou

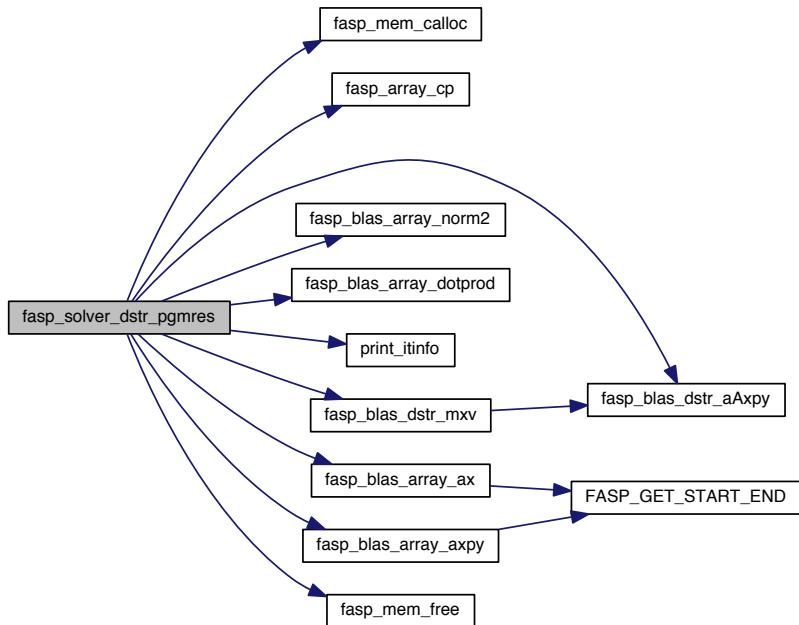
Date

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.

Here is the call graph for this function:

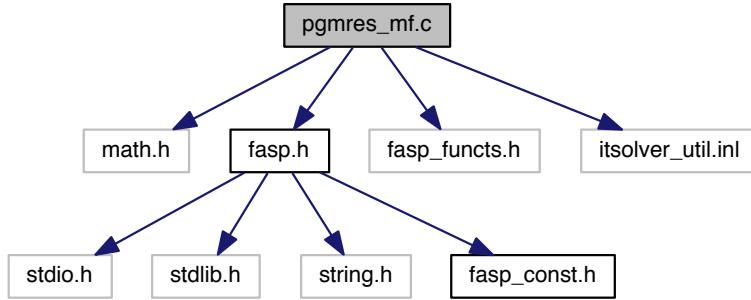


9.66 pgmres_mf.c File Reference

Krylov subspace methods – Preconditioned GMRes (matrix free)

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

Include dependency graph for pgmres_mf.c:



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 print_level)**

Solve "Ax=b" using PGMRES (right preconditioned) iterative method.

9.66.1 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 \leftarrow ES(m) Journal of Computational and Applied Mathematics, 230 (2009) pp. 751-761. UCRL-JRNL-235266.

9.66.2 Function Documentation

- 9.66.2.1 **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 print_level)**

Solve "Ax=b" using PGMRES (right preconditioned) iterative method.

Parameters

<i>mf</i>	Pointer to mxv_matfree : the spmv operation
<i>b</i>	Pointer to dvector : the right hand side
<i>x</i>	Pointer to dvector : the unknowns
<i>pc</i>	Pointer to precond : the structure of precondition

<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>restart</i>	Restarting steps
<i>stop_type</i>	Stopping criteria type – DOES not support this parameter
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Zhiyang Zhou

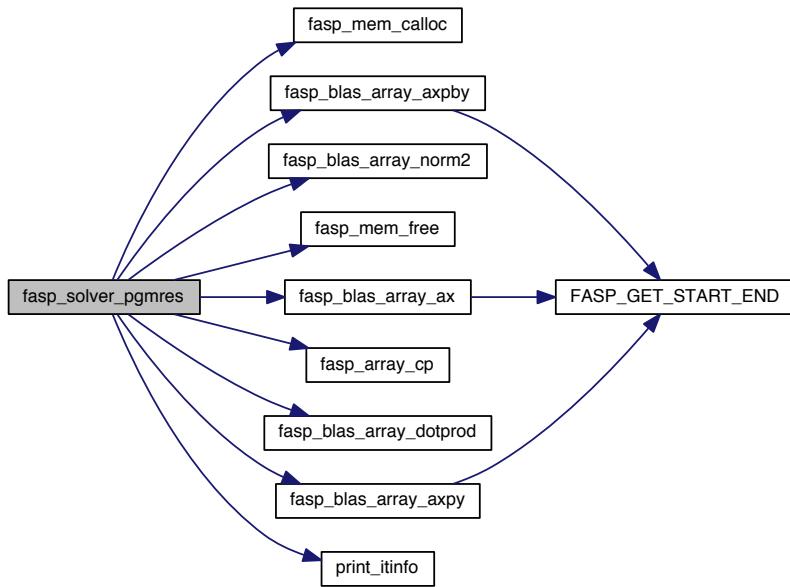
Date

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 51 of file pgmres_mf.c.

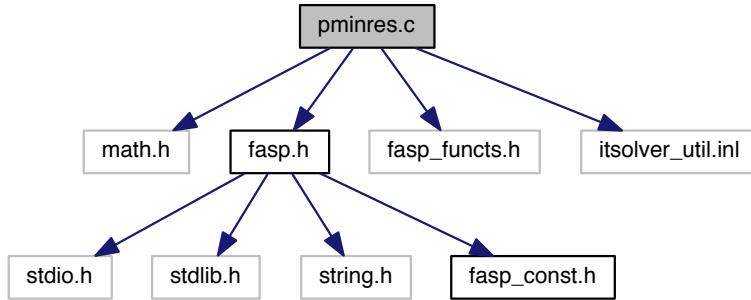
Here is the call graph for this function:



9.67 pminres.c File Reference

Krylov subspace methods – Preconditioned minimal residual.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
Include dependency graph for pminres.c:
```



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 print_level)**
A preconditioned minimal residual (Minres) method for solving $Au=b$.
- **INT fasp_solver_bdcsr_pminres (block_dCSRmat *A, dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT print_level)**
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 print_level)**
A preconditioned minimal residual (Minres) method for solving $Au=b$.

9.67.1 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

$V_k = \text{span}\{r_0, A*r_0, A^2*r_0, \dots, A^{k-1}*r_0\}$,

under some inner product.

For the implementation, we generate a series of $\{p_k\}$ such that $V_k = \text{span}\{p_1, \dots, p_k\}$. 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 $\{p_0, p_1, \dots, p_k\}$;
- prepare for next iteration;
- print the result of k-th iteration; END FOR

Convergence check: $\text{norm}(r)/\text{norm}(b) < \text{tol}$

Stagnation check:

- IF $\text{norm}(\alpha * p_k)/\text{norm}(x_{k+1}) < \text{tol_stag}$
 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 $\text{norm}(r_{k+1})/\text{norm}(b) < \text{tol}$
 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

9.67.2 Function Documentation

9.67.2.1 INT fasp_solver_bdcsr_pminres (**block_dCSRmat * *A*, **dvector** * *b*, **dvector** * *u*, **precond** * *pc*, const **REAL** *tol*, const **INT** *MaxIt*, const **SHORT** *stop_type*, const **SHORT** *print_level*)**

A preconditioned minimal residual (Minres) method for solving $Au=b$.

Parameters

<i>A</i>	Pointer to <code>block_dCSRmat</code> : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to precond: the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Chensong Zhang

Date

05/01/2012

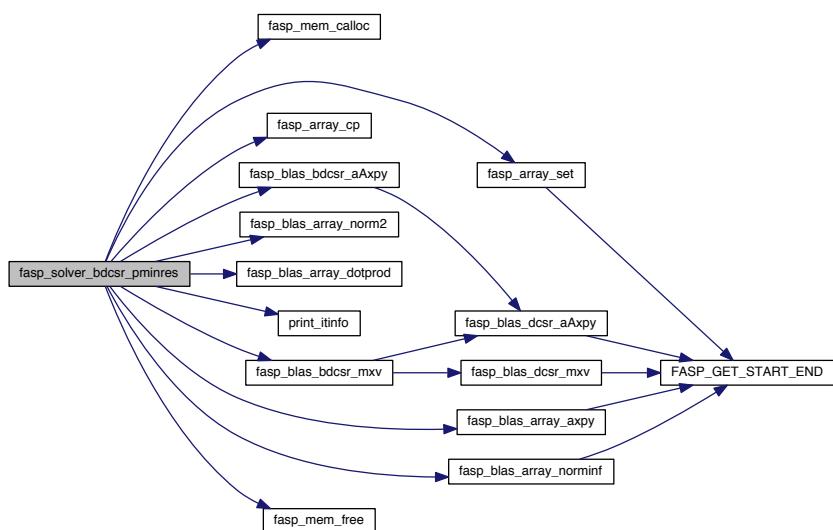
Note

Rewritten based on the original version by Xiaozhe Hu 05/24/2010

Modified by Chensong Zhang on 04/09/2013

Definition at line 500 of file pminres.c.

Here is the call graph for this function:



```
9.67.2.2 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 print_level )
```

A preconditioned minimal residual (Minres) method for solving $Au=b$.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to precond: the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Chensong Zhang

Date

05/01/2012

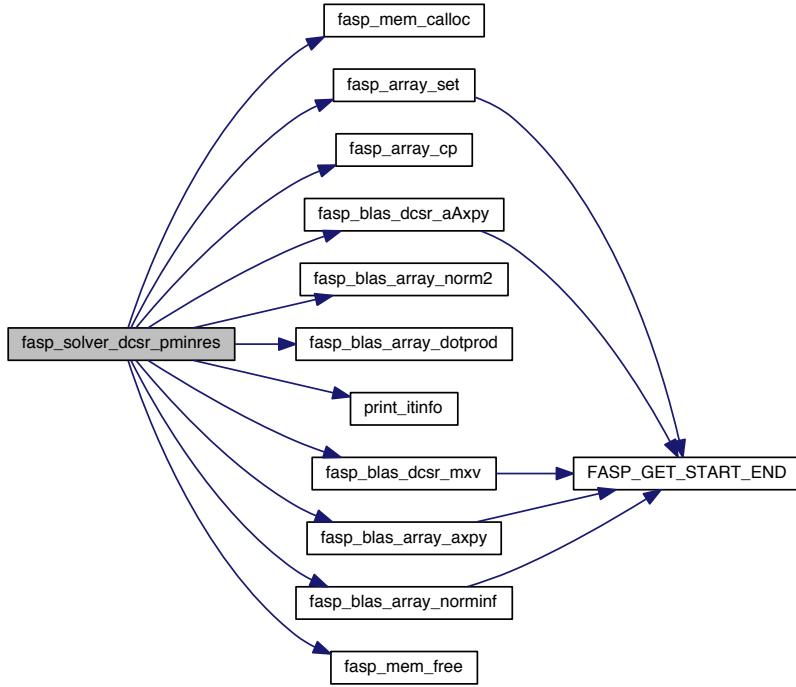
Note

Rewritten based on the original version by Shiquan Zhang 05/10/2010

Modified by Chensong Zhang on 04/09/2013

Definition at line 93 of file pminres.c.

Here is the call graph for this function:



9.67.2.3 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 print_level)

A preconditioned minimal residual (Minres) method for solving $Au=b$.

Parameters

<i>A</i>	Pointer to dSTRmat : the coefficient matrix
<i>b</i>	Pointer to dvector : the right hand side
<i>u</i>	Pointer to dvector : the unknowns
<i>pc</i>	Pointer to precond : the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

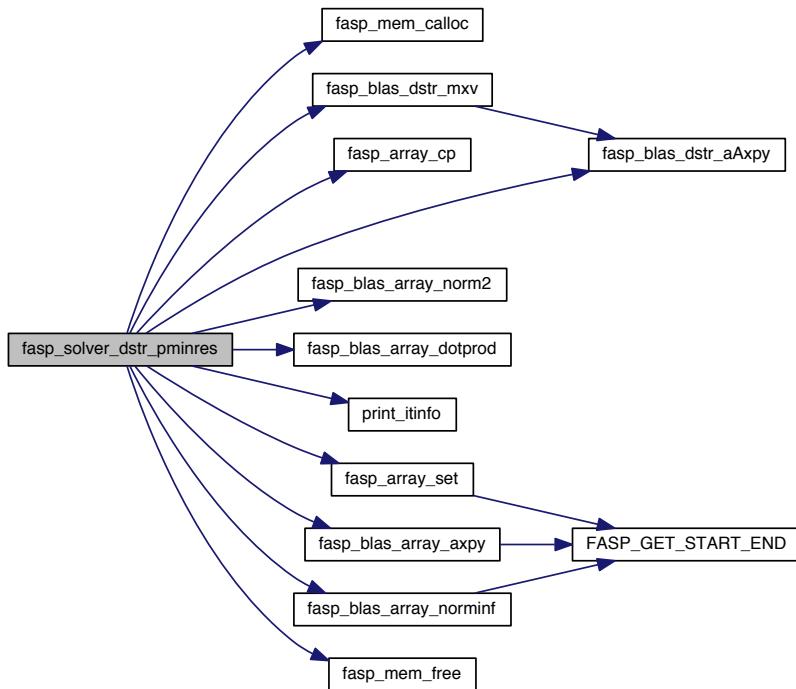
Chensong Zhang

Date

04/09/2013

Definition at line 903 of file pminres.c.

Here is the call graph for this function:

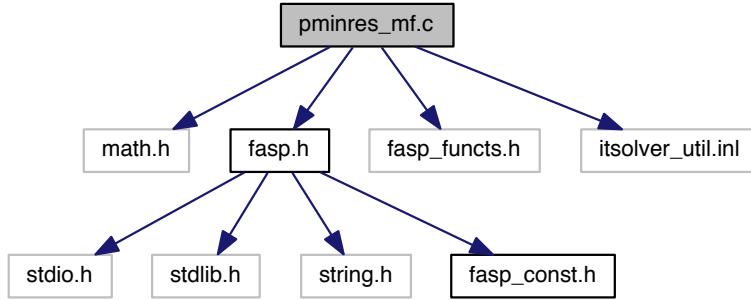


9.68 pminres_mf.c File Reference

Krylov subspace methods – Preconditioned minimal residual (matrix free)

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

Include dependency graph for pminres_mf.c:



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 print_level)`

A preconditioned minimal residual (Minres) method for solving Au=b.

9.68.1 Detailed Description

Krylov subspace methods – Preconditioned minimal residual (matrix free)

Abstract algorithm of Krylov method

Krylov method to solve $A \cdot x = b$ is to generate $\{x_k\}$ to approximate x , where x_k is the optimal solution in Krylov space

$$V_k = \text{span}\{r_0, A \cdot r_0, A^2 \cdot r_0, \dots, A^{k-1} \cdot r_0\},$$

under some inner product.

For the implementation, we generate a series of $\{p_k\}$ such that $V_k = \text{span}\{p_1, \dots, p_k\}$. Details:

Step 0. Given A , b , x_0 , M

Step 1. Compute residual $r_0 = b - A \cdot x_0$ and convergence check;

Step 2. Initialization $z_0 = M^{-1} \cdot r_0$, $p_0 = z_0$;

Step 3. Main loop ...

FOR $k = 0 : \text{MaxIt}$

- get step size $\alpha = f(r_k, z_k, p_k)$;
- update solution: $x_{k+1} = x_k + \alpha \cdot p_k$;
- perform stagnation check;
- update residual: $r_{k+1} = r_k - \alpha \cdot (A \cdot p_k)$;
- perform residual check;

- obtain $p_{\{k+1\}}$ using $\{p_0, p_1, \dots, p_k\}$;
- prepare for next iteration;
- print the result of k-th iteration; END FOR

Convergence check is: $\text{norm}(r)/\text{norm}(b) < \text{tol}$

Stagnation check is like following:

- IF $\text{norm}(\alpha * p_k)/\text{norm}(x_{\{k+1\}}) < \text{tol_stag}$
 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 $\text{norm}(r_{\{k+1\}})/\text{norm}(b) < \text{tol}$
 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

9.68.2 Function Documentation

9.68.2.1 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 *print_level*)

A preconditioned minimal residual (Minres) method for solving $Au=b$.

Parameters

<i>mf</i>	Pointer to mxv_matfree : the spmv operation
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to precond: the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Shiquan Zhang

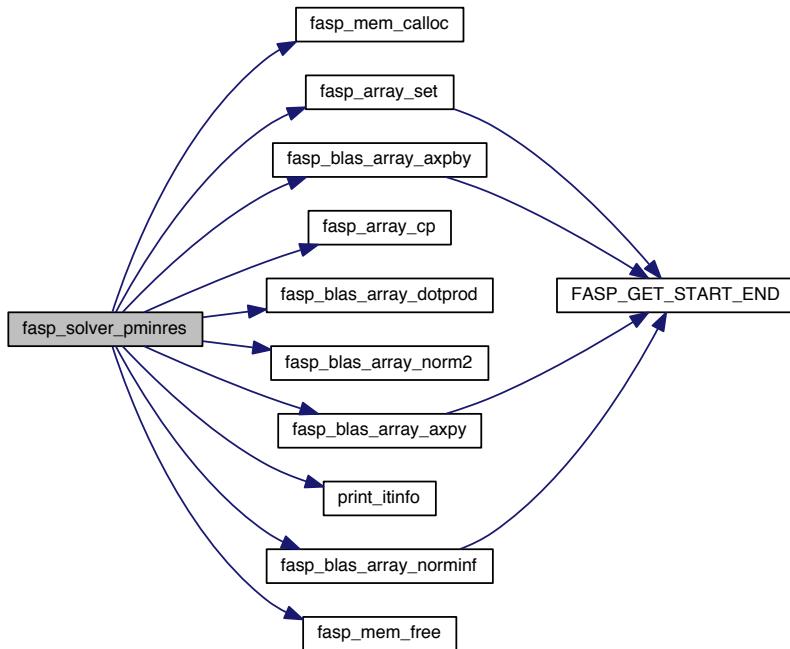
Date

10/24/2010

Rewritten by Chensong Zhang on 05/01/2012 Modified by Feiteng Huang on 09/26/2012: matrix free

Definition at line 90 of file pminres_mf.c.

Here is the call graph for this function:

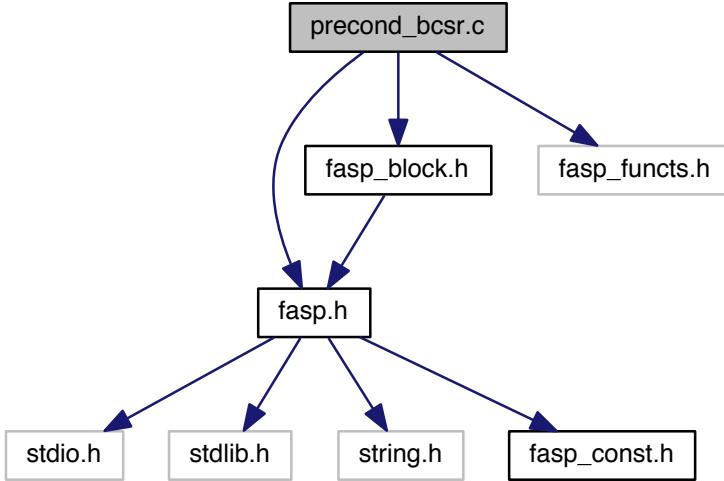


9.69 precond_bcsr.c File Reference

Preconditioners for block CSR matrices.

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

Include dependency graph for precond_bcsr.c:



Functions

- void [fasp_precond_block_diag_3](#) (double *r, double *z, void *data)
block diagonal preconditioning (3x3 block matrix, each diagonal block is solved exactly)
- void [fasp_precond_block_diag_3_amg](#) (double *r, double *z, void *data)
block diagonal preconditioning (3x3 block matrix, each diagonal block is solved by AMG)
- void [fasp_precond_block_diag_4](#) (double *r, double *z, void *data)
block diagonal preconditioning (4x4 block matrix, each diagonal block is solved exactly)
- void [fasp_precond_block_lower_3](#) (double *r, double *z, void *data)
block lower triangular preconditioning (3x3 block matrix, each diagonal block is solved exactly)
- void [fasp_precond_block_lower_3_amg](#) (double *r, double *z, void *data)
block lower triangular preconditioning (3x3 block matrix, each diagonal block is solved by AMG)
- void [fasp_precond_block_lower_4](#) (double *r, double *z, void *data)
block lower triangular preconditioning (4x4 block matrix, each diagonal block is solved exactly)
- void [fasp_precond_sweeping](#) (double *r, double *z, void *data)
sweeping preconditioner for Maxwell equations

9.69.1 Detailed Description

Preconditioners for block CSR matrices.

9.69.2 Function Documentation

9.69.2.1 void fasp_precond_block_diag_3 (double * *r*, double * *z*, void * *data*)

block diagonal preconditioning (3x3 block matrix, each diagonal block is solved exactly)

Parameters

<code>*r</code>	pointer to residual
<code>*z</code>	pointer to preconditioned residual
<code>*data</code>	pointer to precondition data

Author

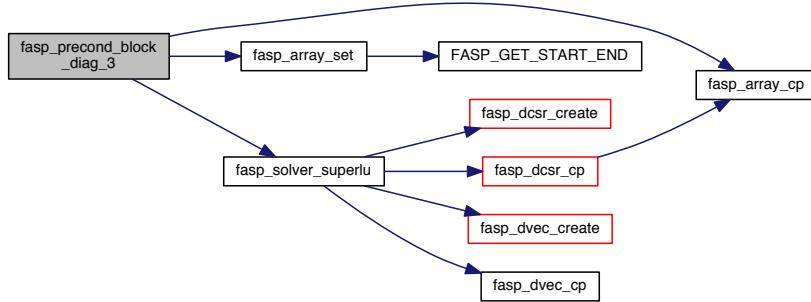
Xiaozhe Hu

Date

07/10/2014

Definition at line 25 of file `precond_bcsr.c`.

Here is the call graph for this function:



9.69.2.2 void `fasp_precond_block_diag_3_amg` (`double * r, double * z, void * data`)

block diagonal preconditioning (3x3 block matrix, each diagonal block is solved by AMG)

Parameters

<code>*r</code>	pointer to residual
<code>*z</code>	pointer to preconditioned residual
<code>*data</code>	pointer to precondition data

Author

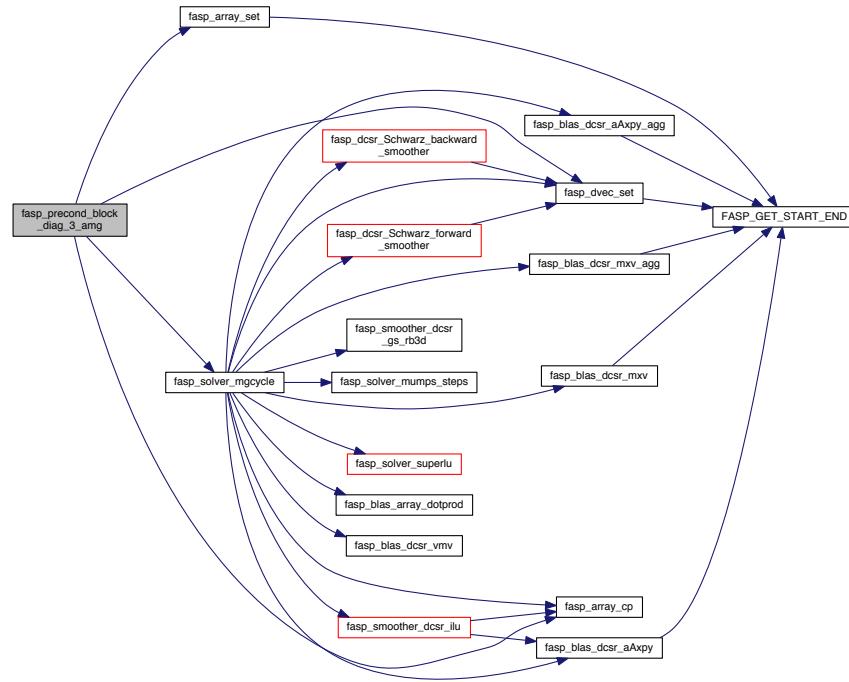
Xiaozhe Hu

Date

07/10/2014

Definition at line 100 of file `precond_bcsr.c`.

Here is the call graph for this function:



9.69.2.3 void fasp_precond_block_diag_4 (double * r, double * z, void * data)

block diagonal preconditioning (4x4 block matrix, each diagonal block is solved exactly)

Parameters

<i>*r</i>	pointer to residual
<i>*z</i>	pointer to preconditioned residual
<i>*data</i>	pointer to precondition data

Author

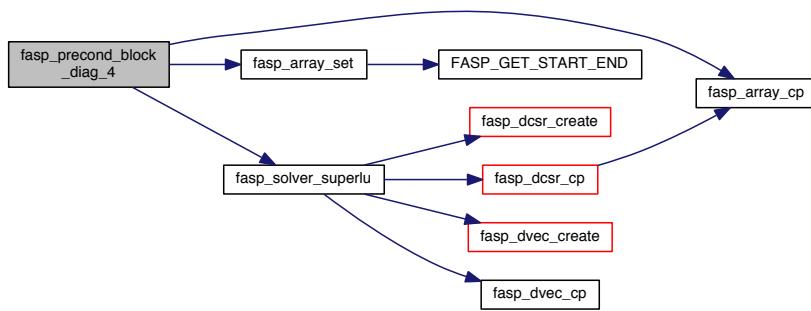
Xiaozhe Hu

Date

07/10/2014

Definition at line 165 of file `precond_bcsr.c`.

Here is the call graph for this function:



9.69.2.4 void `fasp_precond_block_lower_3(double * r, double * z, void * data)`

block lower triangular preconditioning (3x3 block matrix, each diagonal block is solved exactly)

Parameters

<code>*r</code>	pointer to residual
<code>*z</code>	pointer to preconditioned residual
<code>*data</code>	pointer to precondition data

Author

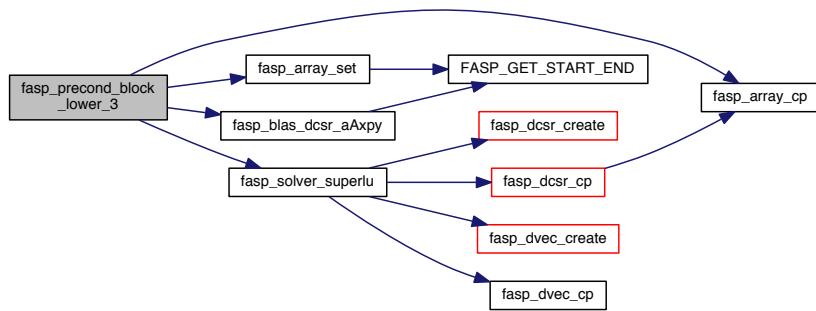
Xiaozhe Hu

Date

07/10/2014

Definition at line 251 of file precondition_bcsr.c.

Here is the call graph for this function:



9.69.2.5 void fasp_precond_block_lower_3_amg (double * r, double * z, void * data)

block lower triangular preconditioning (3x3 block matrix, each diagonal block is solved by AMG)

Parameters

<code>*r</code>	pointer to residual
<code>*z</code>	pointer to preconditioned residual
<code>*data</code>	pointer to precondition data

Author

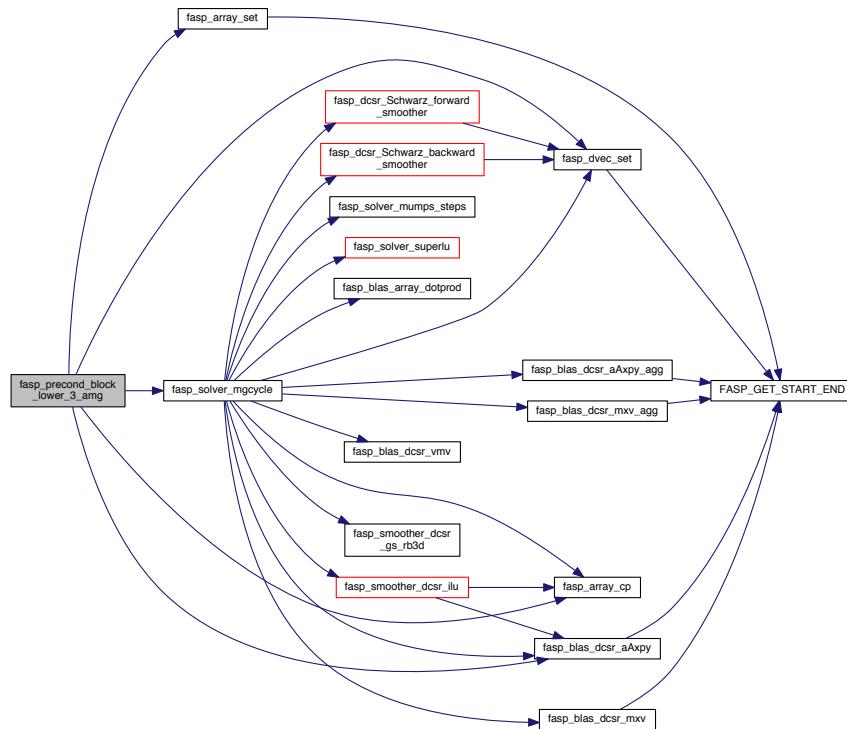
Xiaozhe Hu

Date

07/10/2014

Definition at line 333 of file precondition_bcsr.c.

Here is the call graph for this function:



9.69.2.6 void fasp_precond_block_lower_4(double * r, double * z, void * data)

block lower triangular preconditioning (4x4 block matrix, each diagonal block is solved exactly)

Parameters

*r	pointer to residual
*z	pointer to preconditioned residual
*data	pointer to precondition data

Author

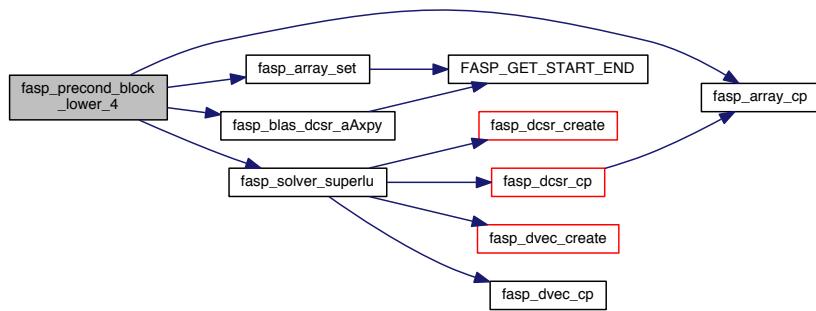
Xiaozhe Hu

Date

07/10/2014

Definition at line 407 of file precondition_bcsr.c.

Here is the call graph for this function:



9.69.2.7 void fasp_precond_sweeping (double * r, double * z, void * data)

sweeping preconditioner for Maxwell equations

Parameters

<code>*r</code>	pointer to residual
<code>*z</code>	pointer to preconditioned residual
<code>*data</code>	pointer to precondition data

Author

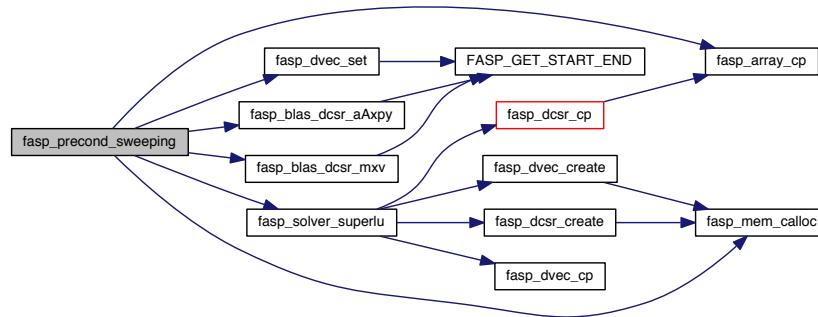
Xiaozhe Hu

Date

05/01/2014

Definition at line 504 of file precondition_bcsr.c.

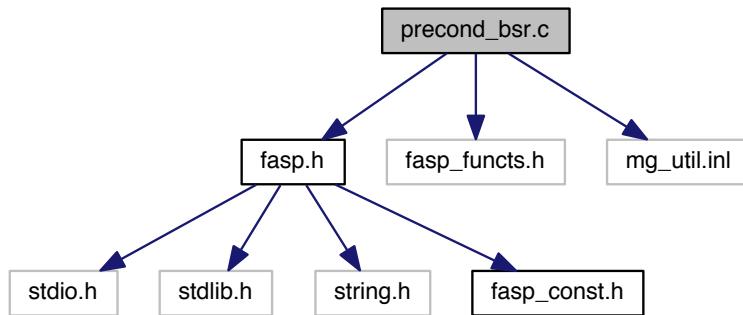
Here is the call graph for this function:



9.70 precondition_bsr.c File Reference

Preconditioners for [dBSRmat](#) matrices.

```
#include "fasp.h"
#include "fasp_functs.h"
#include "mg_util.inl"
Include dependency graph for precondition_bsr.c:
```



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_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.

9.70.1 Detailed Description

Preconditioners for [dBSRmat](#) matrices.

9.70.2 Function Documentation

9.70.2.1 void [fasp_precond_dbsr_amg](#) (**REAL** * *r*, **REAL** * *z*, void * *data*)

AMG preconditioner.

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
<i>z</i>	Pointer to preconditioned vector
<i>data</i>	Pointer to precondition data

Author

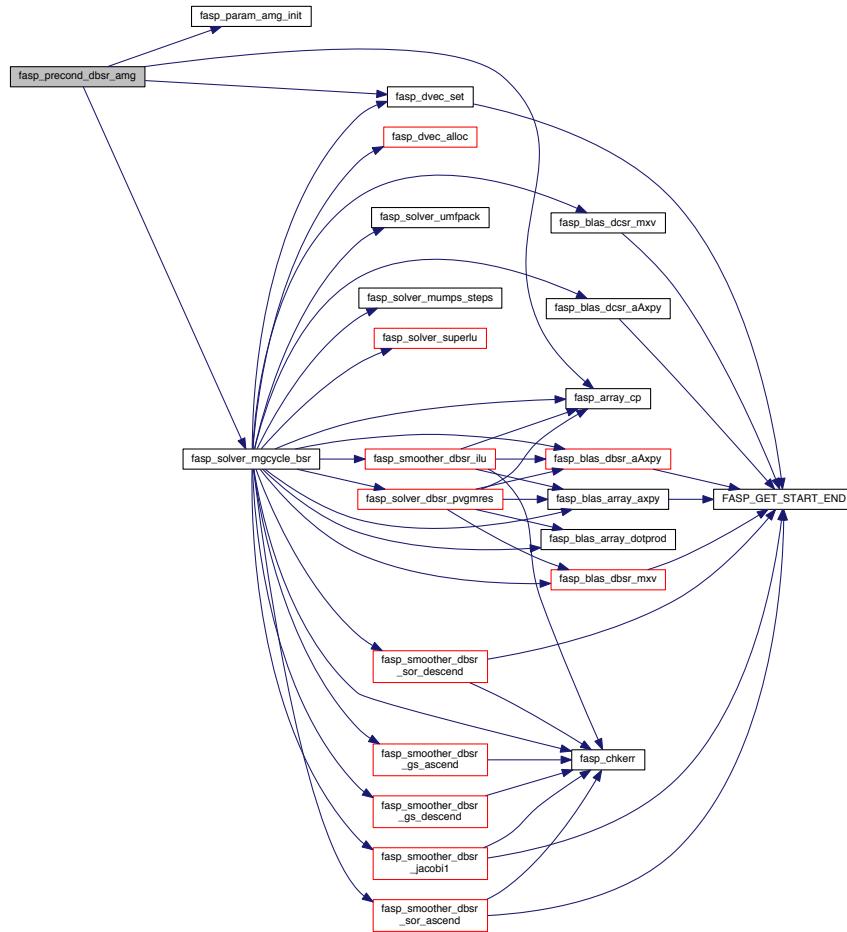
Xiaozhe Hu

Date

08/07/2011

Definition at line 563 of file precondition_bsr.c.

Here is the call graph for this function:



9.70.2.2 void fasp_precond_dbsr_amg_nk (REAL * r, REAL * z, void * data)

AMG with extra near kernel solve preconditioner.

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
<i>z</i>	Pointer to preconditioned vector

<code>data</code>	Pointer to precondition data
-------------------	------------------------------

Author

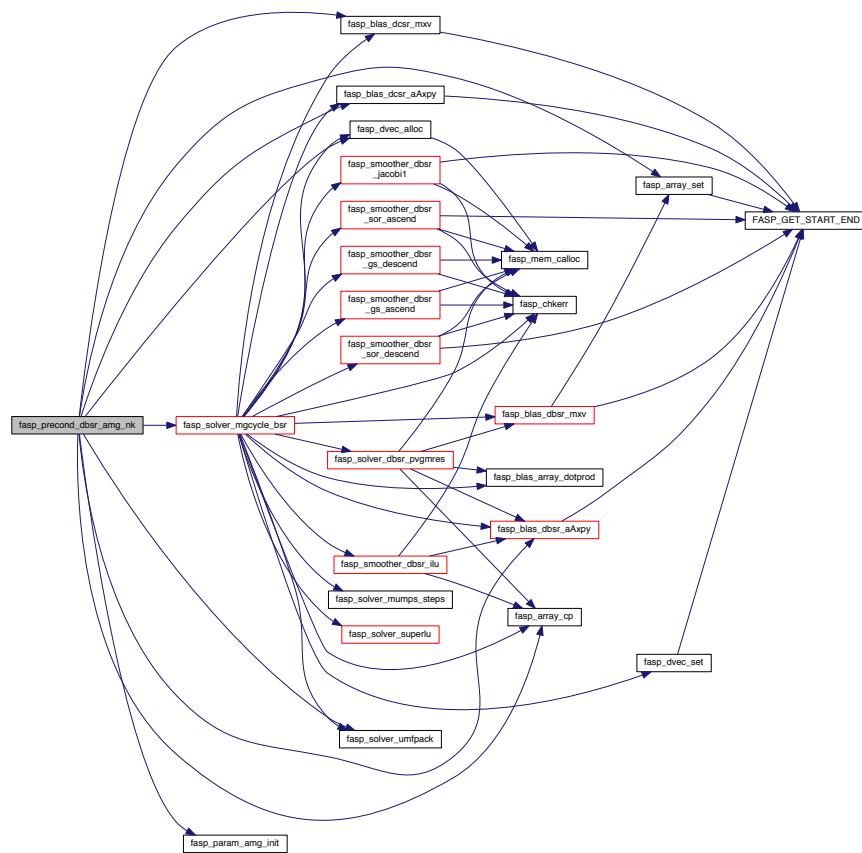
Xiaozhe Hu

Date

05/26/2014

Definition at line 643 of file precondition_bsr.c.

Here is the call graph for this function:



9.70.2.3 void fasp_precond_dbsr_diag (REAL * r, REAL * z, void * data)

Diagonal preconditioner $z = \text{inv}(D) \cdot r$.

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
<i>z</i>	Pointer to preconditioned vector
<i>data</i>	Pointer to precondition data

Author

Zhou Zhiyang, Xiaozhe Hu

Date

10/26/2010

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

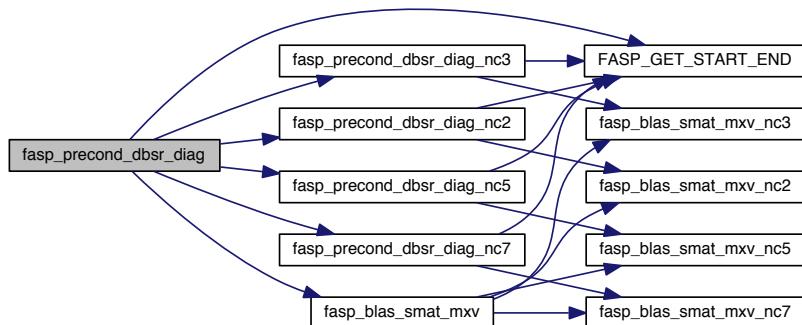
05/24/2012

Note

Works for general nb (Xiaozhe)

Definition at line 37 of file precond_bsr.c.

Here is the call graph for this function:



9.70.2.4 void fasp_precond_dbsr_diag_nc2 (REAL * *r*, REAL * *z*, void * *data*)

Diagonal preconditioner $z = \text{inv}(D) * r$.

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
<i>z</i>	Pointer to preconditioned vector
<i>data</i>	Pointer to precondition data

Author

Zhou Zhiyang, XIAOZHE HU

Date

11/18/2011

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

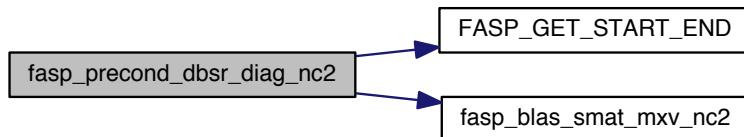
05/24/2012

Note

Works for 2-component (XIAOZHE)

Definition at line 111 of file precondition_bsr.c.

Here is the call graph for this function:



9.70.2.5 void fasp_precond_dbsr_diag_nc3 (REAL * r, REAL * z, void * data)

Diagonal preconditioner $z = \text{inv}(D) * r$.

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
<i>z</i>	Pointer to preconditioned vector
<i>data</i>	Pointer to precondition data

Author

Zhou Zhiyang, XIAOZHE HU

Date

01/06/2011

Modified by Chunsheng Feng Xiaoqiang Yue

Date

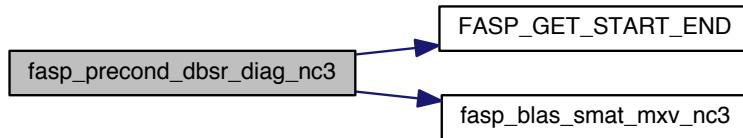
05/24/2012

Note

Works for 3-component (Xiaozhe)

Definition at line 161 of file precond_bsr.c.

Here is the call graph for this function:



9.70.2.6 void fasp_precond_dbsr_diag_nc5 (REAL * r, REAL * z, void * data)

Diagonal preconditioner $z = \text{inv}(D) * r$.

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
<i>z</i>	Pointer to preconditioned vector
<i>data</i>	Pointer to precondition data

Author

Zhou Zhiyang, Xiaozhe Hu

Date

01/06/2011

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

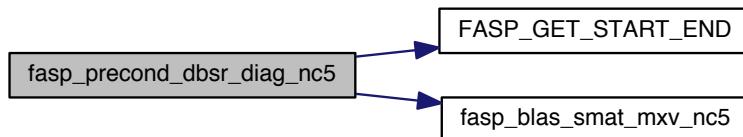
05/24/2012

Note

Works for 5-component (Xiaozhe)

Definition at line 211 of file precondition_bsr.c.

Here is the call graph for this function:



9.70.2.7 void fasp_precond_dbsr_diag_nc7(REAL * r, REAL * z, void * data)

Diagonal preconditioner $z = \text{inv}(D) * r$.

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
<i>z</i>	Pointer to preconditioned vector
<i>data</i>	Pointer to precondition data

Author

Zhou Zhiyang, Xiaozhe Hu

Date

01/06/2011

Modified by Chunsheng Feng Xiaoqiang Yue

Date

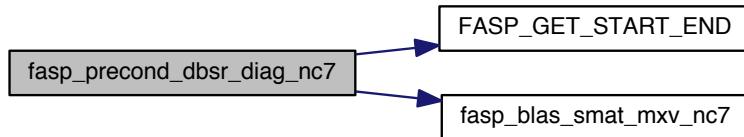
05/24/2012

Note

Works for 7-component (Xiaozhe)

Definition at line 260 of file precond_bsr.c.

Here is the call graph for this function:



9.70.2.8 void fasp_precond_dbsr_ilu (**REAL** * *r*, **REAL** * *z*, **void** * *data*)

ILU preconditioner.

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
<i>z</i>	Pointer to preconditioned vector
<i>data</i>	Pointer to precondition data

Author

Shiquan Zhang, Xiaozhe Hu

Date

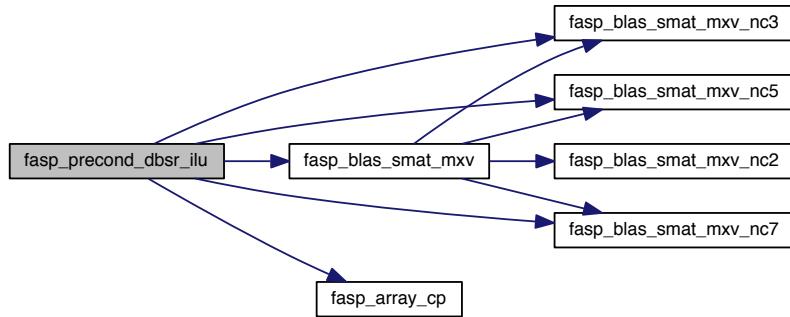
11/09/2010

Note

Works for general nb (Xiaozhe)

Definition at line 306 of file precondition_bsr.c.

Here is the call graph for this function:



9.70.2.9 void fasp_precond_dbsr_nl_amli (REAL * r, REAL * z, void * data)

Nonlinear AMLI-cycle AMG preconditioner.

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
<i>z</i>	Pointer to preconditioned vector
<i>data</i>	Pointer to precondition data

Author

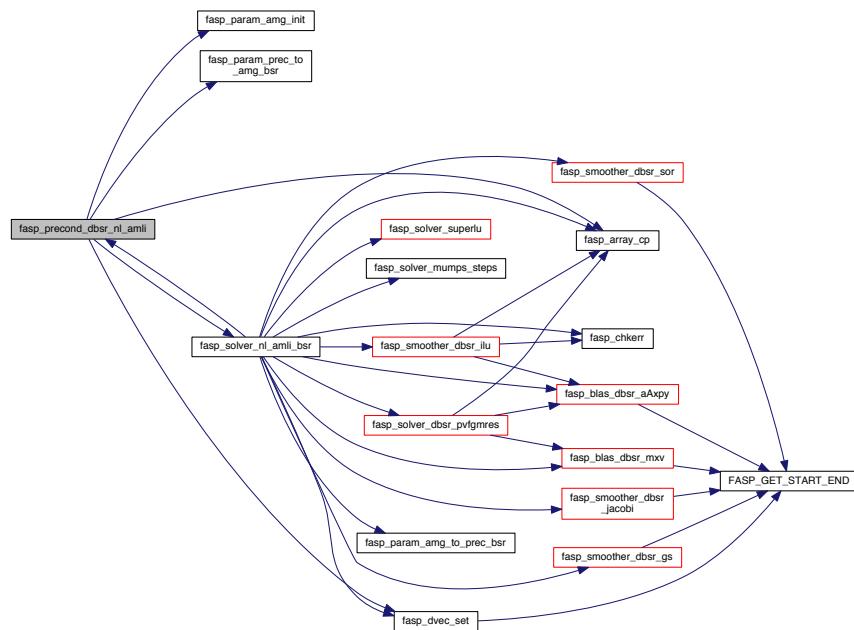
Xiaozhe Hu

Date

02/06/2012

Definition at line 607 of file precond_bsr.c.

Here is the call graph for this function:

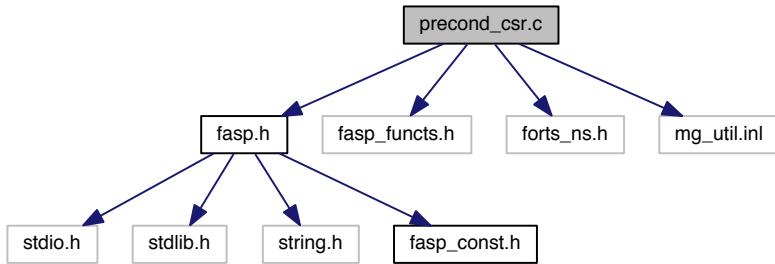


9.71 precond_csr.c File Reference

Preconditioners for [dCSRmat](#) matrices.

```
#include "fasp.h"
#include "fasp_functs.h"
#include "forts_ns.h"
#include "mg_util.inl"
```

Include dependency graph for precond_csr.c:



Functions

- `precond * fasp_precond_setup (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 (SHORT precond_type, precond *pc)`
free preconditioner

9.71.1 Detailed Description

Preconditioners for `dCSRmat` matrices.

9.71.2 Function Documentation

9.71.2.1 void fasp_precond_amg (**REAL** * *r*, **REAL** * *z*, **void** * *data*)

AMG preconditioner.

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
<i>z</i>	Pointer to preconditioned vector
<i>data</i>	Pointer to precondition data

Author

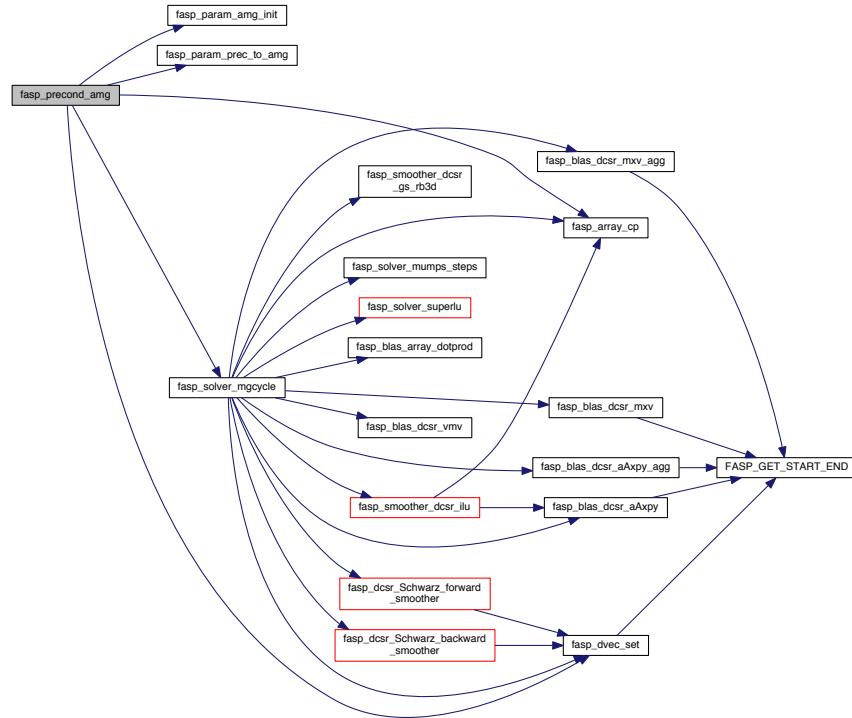
Chensong Zhang

Date

04/06/2010

Definition at line 400 of file precond_csr.c.

Here is the call graph for this function:



9.71.2.2 void fasp_precond_amg_nk (**REAL** * *r*, **REAL** * *z*, **void** * *data*)

AMG with extra near kernel solve as preconditioner.

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
<i>z</i>	Pointer to preconditioned vector
<i>data</i>	Pointer to precondition data

Author

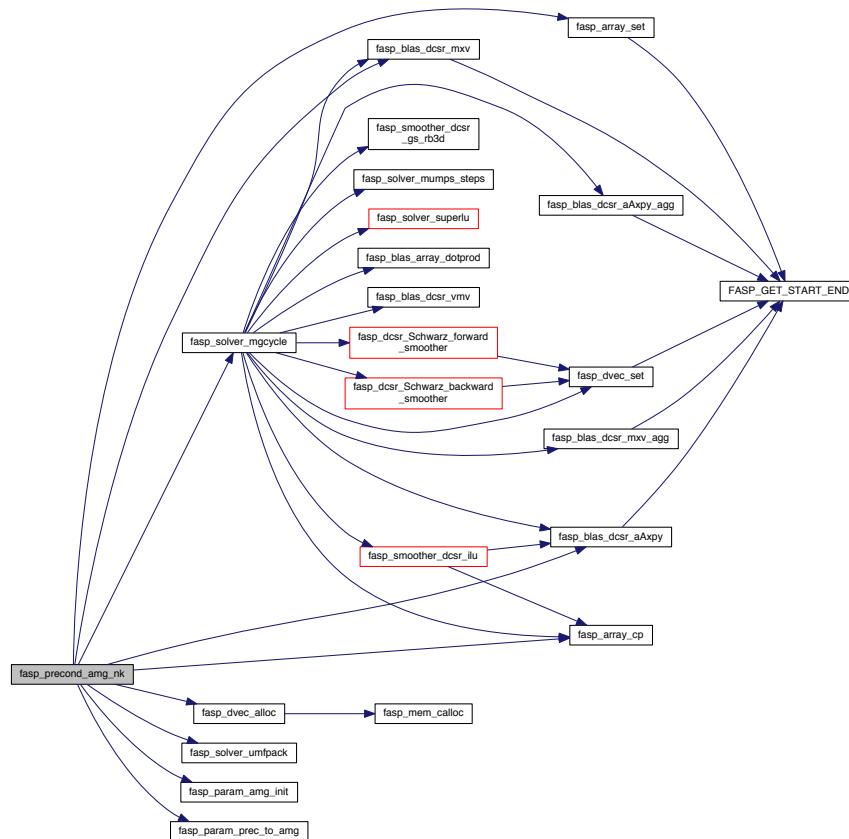
Xiaozhe Hu

Date

05/26/2014

Definition at line 535 of file precondition_csr.c.

Here is the call graph for this function:

**9.71.2.3 void fasp_precond_amli (REAL * r, REAL * z, void * data)**

AMLI AMG preconditioner.

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
<i>z</i>	Pointer to preconditioned vector
<i>data</i>	Pointer to precondition data

Author

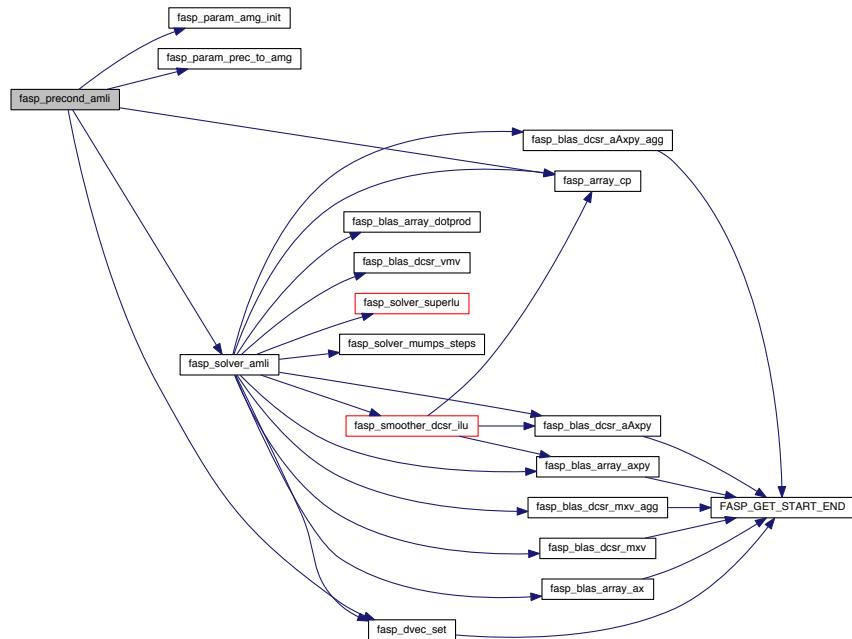
Xiaozhe Hu

Date

01/23/2011

Definition at line 469 of file precond_csr.c.

Here is the call graph for this function:



9.71.2.4 void fasp_precond_diag (REAL * r, REAL * z, void * data)

Diagonal preconditioner $z = \text{inv}(D) * r$.

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
----------	---

<i>z</i>	Pointer to preconditioned vector
<i>data</i>	Pointer to precondition data

Author

Chensong Zhang

Date

04/06/2010

Definition at line 159 of file precond_csr.c.

9.71.2.5 void fasp_precond_famg (**REAL * *r*, **REAL** * *z*, void * *data*)**

Full AMG preconditioner.

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
<i>z</i>	Pointer to preconditioned vector
<i>data</i>	Pointer to precondition data

Author

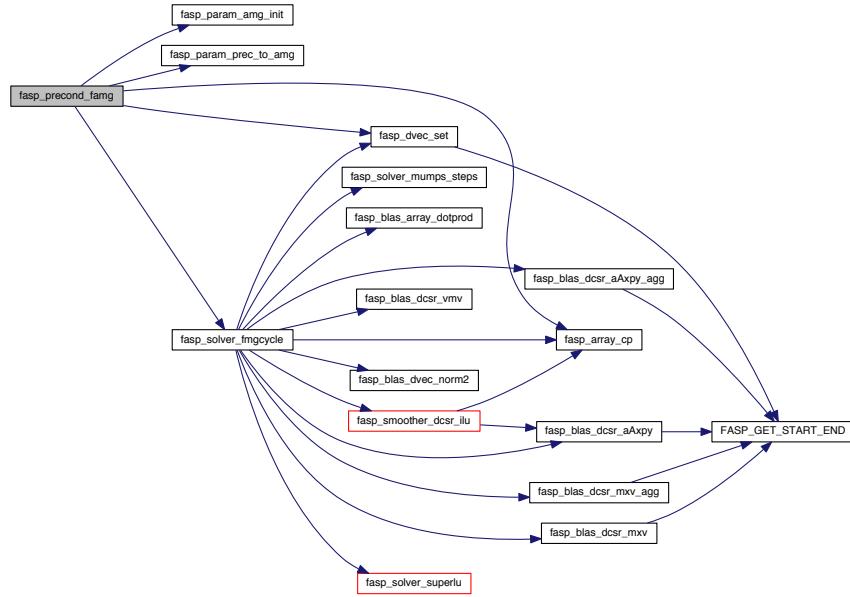
Xiaozhe Hu

Date

02/27/2011

Definition at line 436 of file precond_csr.c.

Here is the call graph for this function:



9.71.2.6 void fasp_precond_free (**SHORT** *precond_type*, *precond* * *pc*)

free preconditioner

Parameters

<i>precond_type</i>	Preconditioner type
* <i>pc</i>	precondition data & fct

Returns

void

Author

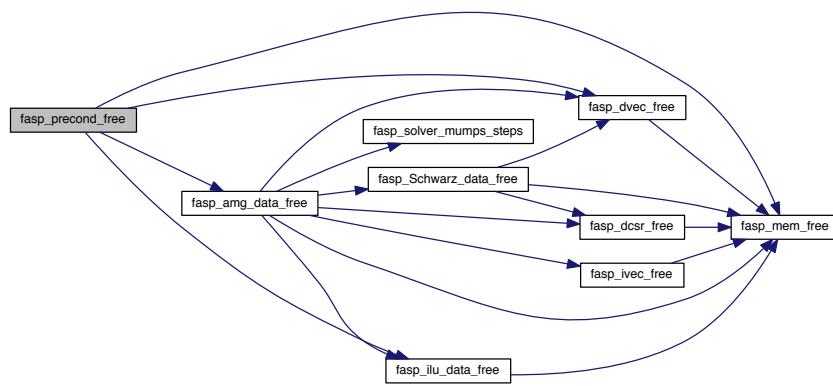
Feiteng Huang

Date

12/24/2012

Definition at line 619 of file precondition_csr.c.

Here is the call graph for this function:

**9.71.2.7 void fasp_precond_ilu (REAL * r, REAL * z, void * data)**

ILU preconditioner.

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
<i>z</i>	Pointer to preconditioned vector
<i>data</i>	Pointer to precondition data

Author

Shiquan Zhang

Date

04/06/2010

Definition at line 185 of file precondition_csr.c.

Here is the call graph for this function:



9.71.2.8 void fasp_precond_ilu_backward (REAL * r, REAL * z, void * data)

ILU preconditioner: only backward sweep.

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
<i>z</i>	Pointer to preconditioned vector
<i>data</i>	Pointer to precondition data

Author

Xiaozhe Hu, Shiquan Zhang

Date

04/06/2010

Definition at line 302 of file precond_csr.c.

Here is the call graph for this function:



9.71.2.9 void fasp_precond_ilu_forward (REAL * r, REAL * z, void * data)

ILU preconditioner: only forward sweep.

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
<i>z</i>	Pointer to preconditioned vector
<i>data</i>	Pointer to precondition data

Author

Xiaozhe Hu, Shiquang Zhang

Date

04/06/2010

Definition at line 249 of file precond_csr.c.

Here is the call graph for this function:



9.71.2.10 void fasp_precond_nl_amli (REAL * r, REAL * z, void * data)

Nonlinear AMLI AMG preconditioner.

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
<i>z</i>	Pointer to preconditioned vector
<i>data</i>	Pointer to precondition data

Author

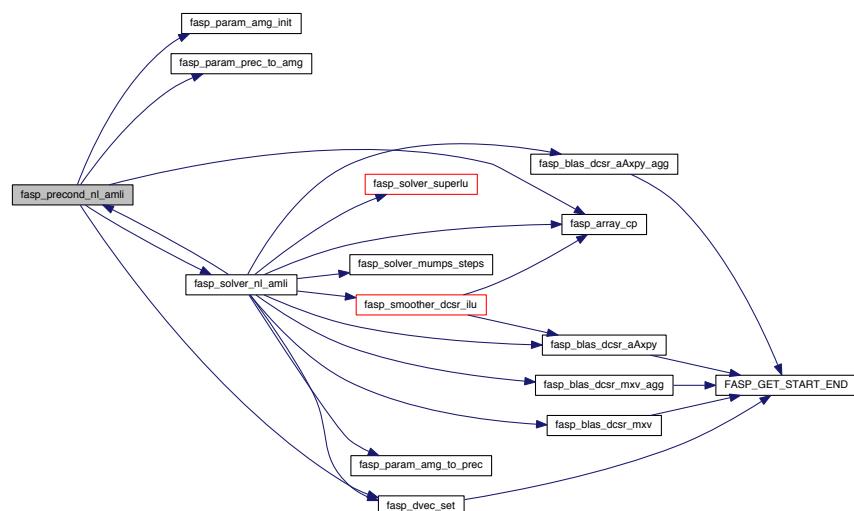
Xiaozhe Hu

Date

04/25/2011

Definition at line 502 of file precondition_csr.c.

Here is the call graph for this function:



9.71.2.11 void fasp_precond_Schwarz (REAL * r, REAL * z, void * data)

get z from r by Schwarz

Parameters

<code>*r</code>	pointer to residual
<code>*z</code>	pointer to preconditioned residual
<code>*data</code>	pointer to precondition data

Author

Xiaozhe Hu

Date

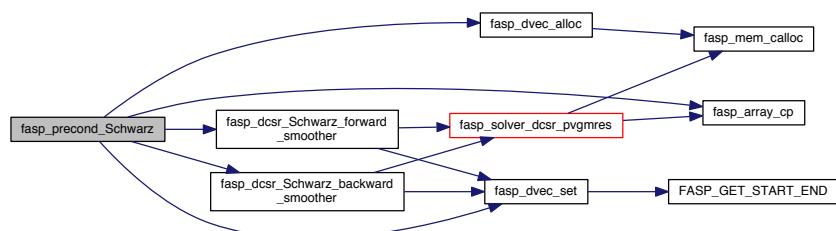
03/22/2010

Note

Change Schwarz interface by Zheng Li on 11/18/2014

Definition at line 355 of file precond_csr.c.

Here is the call graph for this function:



9.71.2.12 precond * fasp_precond_setup (SHORT precond_type, AMG_param * amgparam, ILU_param * iluparam, dCSRmat * A)

Setup preconditioner interface for iterative methods.

Parameters

<code>precond_type</code>	Preconditioner type
<code>*amgparam</code>	AMG parameters
<code>*iluparam</code>	ILU parameters
<code>*A</code>	Pointer to coefficient matrix

Returns

Pointer to preconditioner

Author

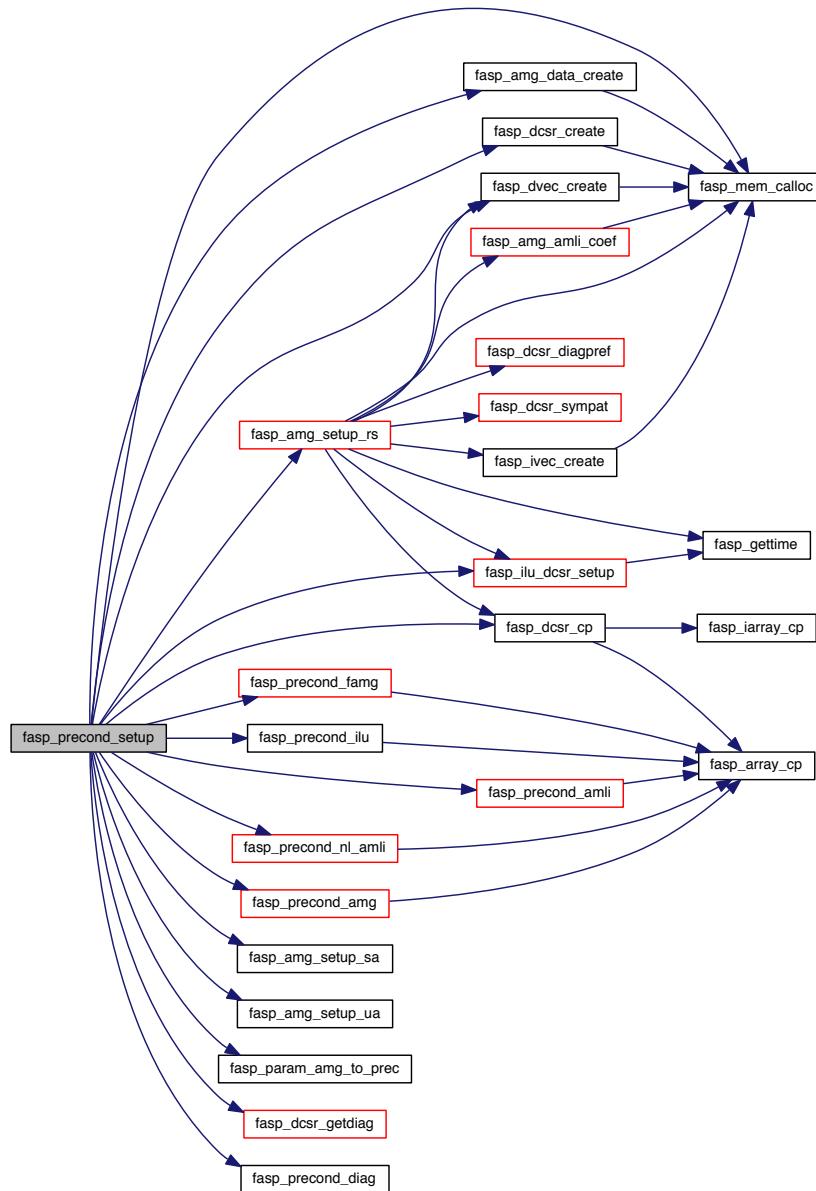
Feiteng Huang

Date

05/18/2009

Definition at line 32 of file precondition_csr.c.

Here is the call graph for this function:

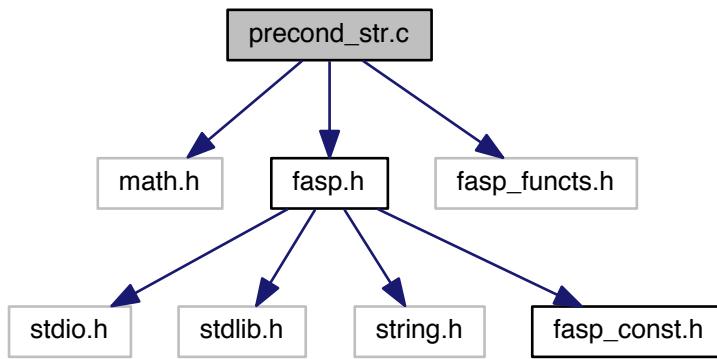


9.72 precond_str.c File Reference

Preconditioners for `dSTRmat` matrices.

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

Include dependency graph for `precond_str.c`:



Functions

- void `fasp_precond_dstr_diag (REAL *r, REAL *z, void *data)`
*Diagonal preconditioner $z = \text{inv}(D) * r$.*
- 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)

9.72.1 Detailed Description

Preconditioners for `dSTRmat` matrices.

9.72.2 Function Documentation

9.72.2.1 void fasp_precond_dstr_blockgs (**REAL** * *r*, **REAL** * *z*, **void** * *data*)

CPR-type preconditioner (STR format)

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
<i>z</i>	Pointer to preconditioned vector
<i>data</i>	Pointer to precondition data

Author

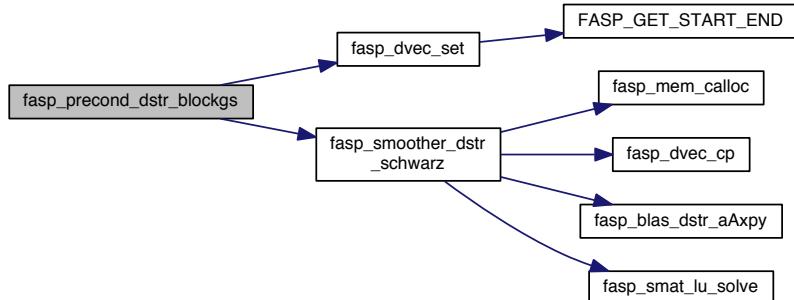
Shiquan Zhang

Date

10/17/2010

Definition at line 1707 of file precondition_str.c.

Here is the call graph for this function:



9.72.2.2 void fasp_precond_dstr_diag (**REAL** * *r*, **REAL** * *z*, **void** * *data*)

Diagonal preconditioner $z = \text{inv}(D) * r$.

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
<i>z</i>	Pointer to preconditioned vector
<i>data</i>	Pointer to precondition data

Author

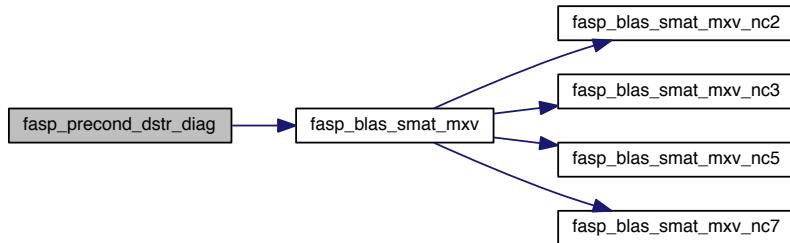
Shiquan Zhang

Date

04/06/2010

Definition at line 27 of file precond_str.c.

Here is the call graph for this function:

**9.72.2.3 void fasp_precond_dstr_ilu0 (REAL * r, REAL * z, void * data)**

Preconditioning using STR_ILU(0) decomposition.

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
<i>z</i>	Pointer to preconditioned vector
<i>data</i>	Pointer to precondition data

Author

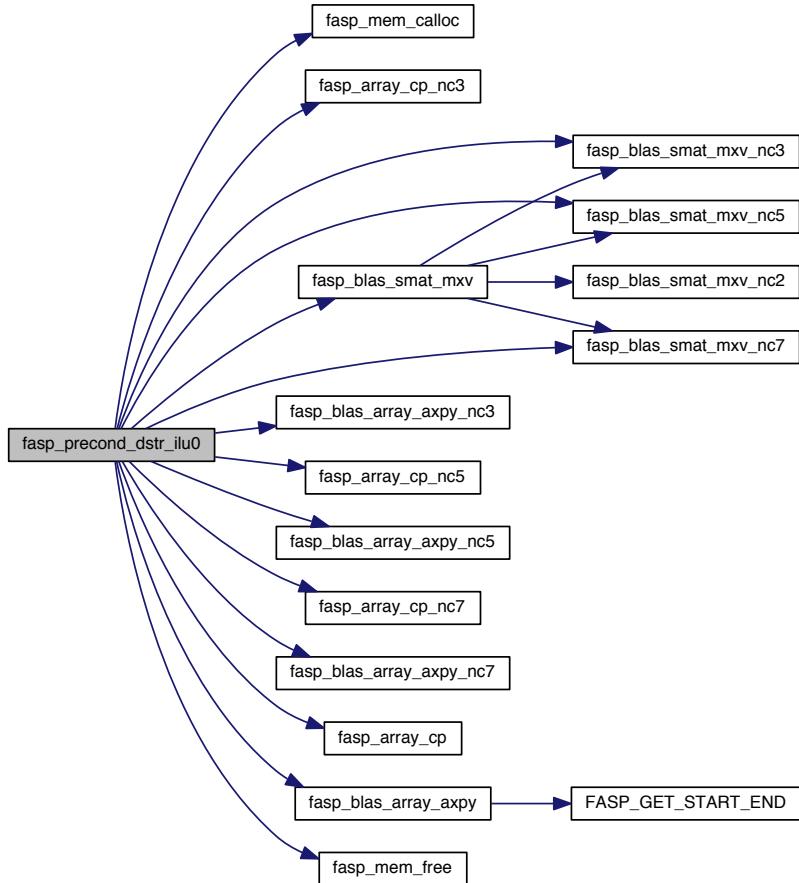
Shiquan Zhang

Date

04/21/2010

Definition at line 55 of file precondition_str.c.

Here is the call graph for this function:



9.72.2.4 void fasp_precond_dstr_ilu0_backward (REAL * r, REAL * z, void * data)

Preconditioning using STR_ILU(0) decomposition: Uz = r.

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
<i>z</i>	Pointer to preconditioned vector

<code>data</code>	Pointer to precondition data
-------------------	------------------------------

Author

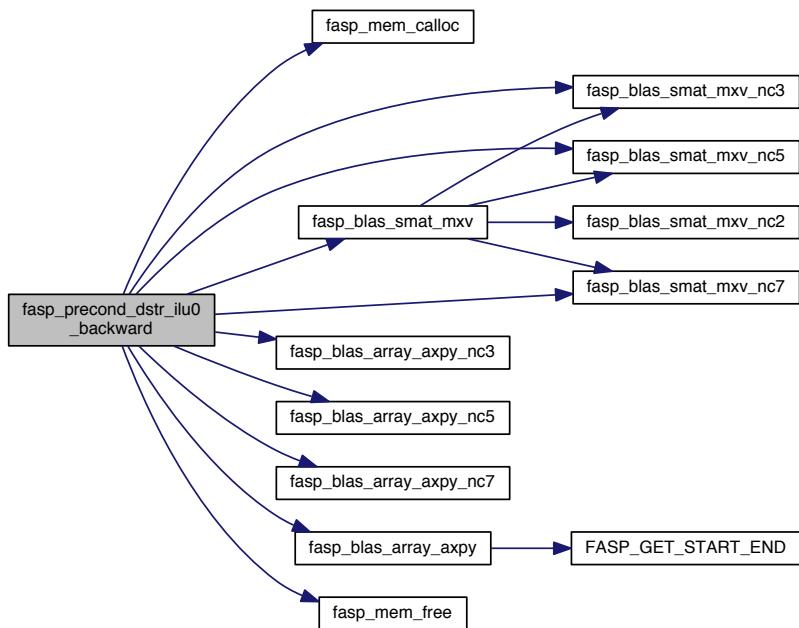
Shiquan Zhang

Date

06/07/2010

Definition at line 979 of file `precond_str.c`.

Here is the call graph for this function:



9.72.2.5 void `fasp_precond_dstr_ilu0_forward` (`REAL * r, REAL * z, void * data`)

Preconditioning using STR_ILU(0) decomposition: $Lz = r$.

Parameters

<code>r</code>	Pointer to the vector needs preconditioning
<code>z</code>	Pointer to preconditioned vector
<code>data</code>	Pointer to precondition data

Author

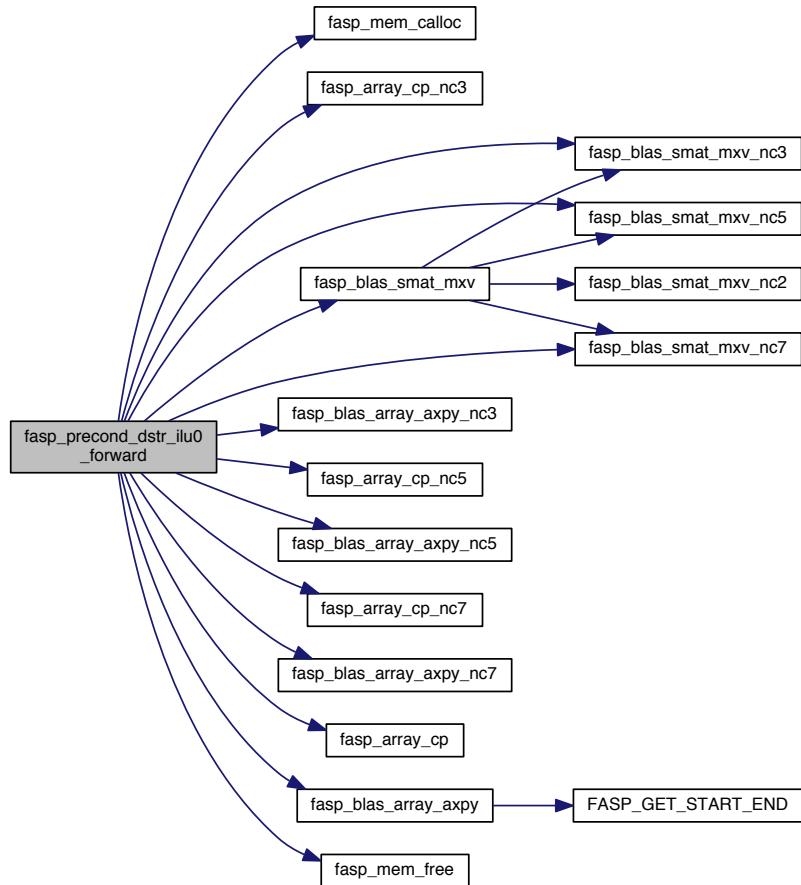
Shiquan Zhang

Date

06/07/2010

Definition at line 816 of file precondition_str.c.

Here is the call graph for this function:



9.72.2.6 void fasp_precond_dstr_ilu1 (REAL * r, REAL * z, void * data)

Preconditioning using STR_ILU(1) decomposition.

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
<i>z</i>	Pointer to preconditioned vector

<i>data</i>	Pointer to precondition data
-------------	------------------------------

Author

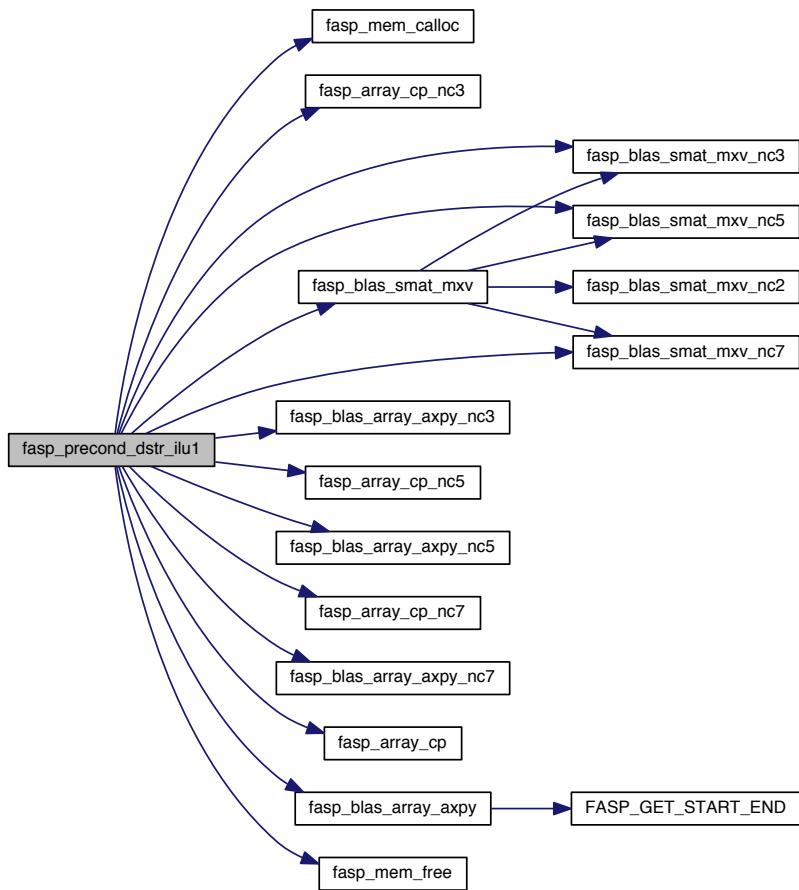
Shiquan Zhang

Date

04/21/2010

Definition at line 337 of file precond_str.c.

Here is the call graph for this function:



9.72.2.7 void fasp_precond_dstr_ilu1_backward (REAL * r, REAL * z, void * data)

Preconditioning using STR_ILU(1) decomposition: Uz = r.

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
<i>z</i>	Pointer to preconditioned vector
<i>data</i>	Pointer to precondition data

Author

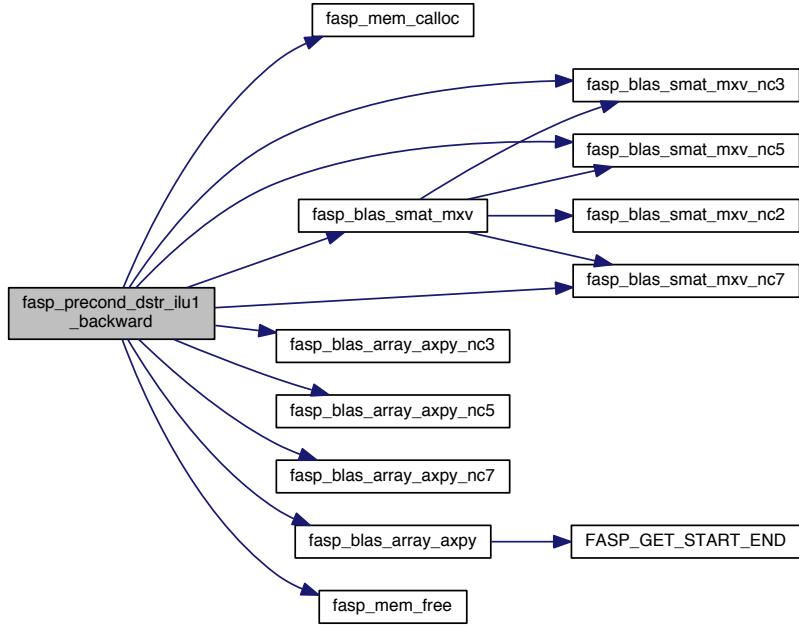
Shiquan Zhang

Date

04/21/2010

Definition at line 1426 of file precondition_str.c.

Here is the call graph for this function:

**9.72.2.8 void fasp_precond_dstr_ilu1_forward (REAL * r, REAL * z, void * data)**

Preconditioning using STR_ILU(1) decomposition: Lz = r.

Parameters

<i>r</i>	Pointer to the vector needs preconditioning
<i>z</i>	Pointer to preconditioned vector
<i>data</i>	Pointer to precondition data

Author

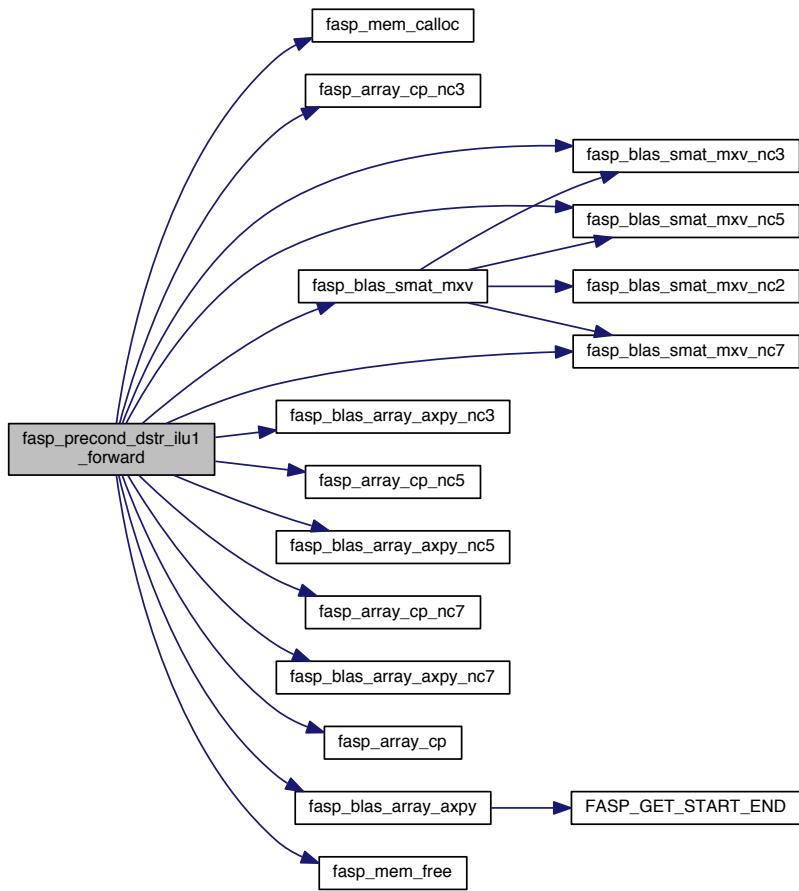
Shiquan Zhang

Date

04/21/2010

Definition at line 1160 of file precond_str.c.

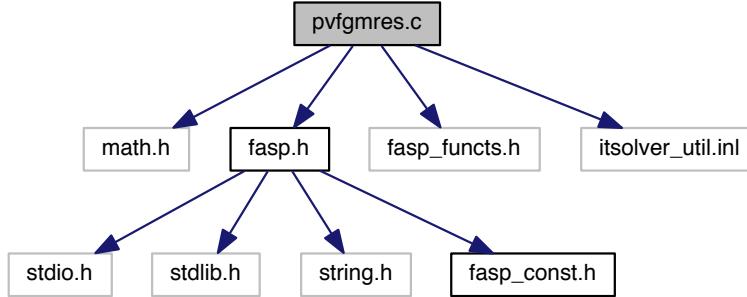
Here is the call graph for this function:



9.73 pfgmres.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"
Include dependency graph for pvfgmres.c:
```



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 print_level)`
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 print_level)`
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_bdcsr_pvfgmres (block_dCSRmat *A, dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT print_level)`
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.

9.73.1 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 \leftarrow ES(m) Journal of Computational and Applied Mathematics, 230 (2009) pp. 751-761. UCRL-JRNL-235266.
 This file is modified from [pvgmres.c](#)

9.73.2 Function Documentation

9.73.2.1 **INT fasp_solver_bdcsr_pvfgmres (*block_dCSRmat* * *A*, *dvector* * *b*, *dvector* * *x*, *precond* * *pc*, const **REAL** *tol*, const **INT** *MaxIt*, const **SHORT** *restart*, const **SHORT** *stop_type*, const **SHORT** *print_level*)**

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

<i>*A</i>	pointer to the coefficient matrix
<i>*b</i>	pointer to the right hand side vector
<i>*x</i>	pointer to the solution vector
<i>MaxIt</i>	maximal iteration number allowed
<i>tol</i>	tolerance
<i>*pc</i>	pointer to preconditioner data
<i>print_level</i>	how much of the SOLVE-INFORMATION be printed
<i>stop_type</i>	default stopping criterion,i.e. $\ r_k\ /\ r_0\ < tol$, is used.
<i>restart</i>	number of restart for GMRES

Returns

number of iteration if succeed

Author

Xiaozhe Hu

Date

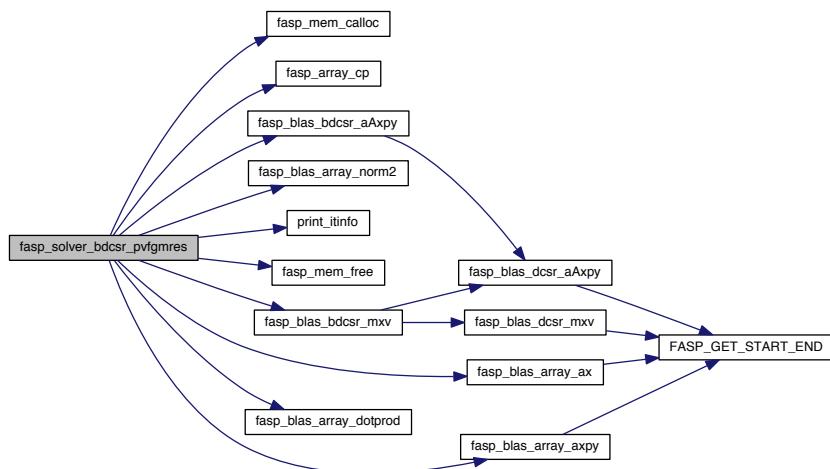
01/04/2012

Note

Based on Zhiyang Zhou's [pvgmres.c](#)

Definition at line 678 of file pvfgmres.c.

Here is the call graph for this function:



9.73.2.2 **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** *print_level*)**

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

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>x</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to precond: the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>restart</i>	Restarting steps
<i>stop_type</i>	Stopping criteria type – DOES not support this parameter
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Xiaozhe Hu

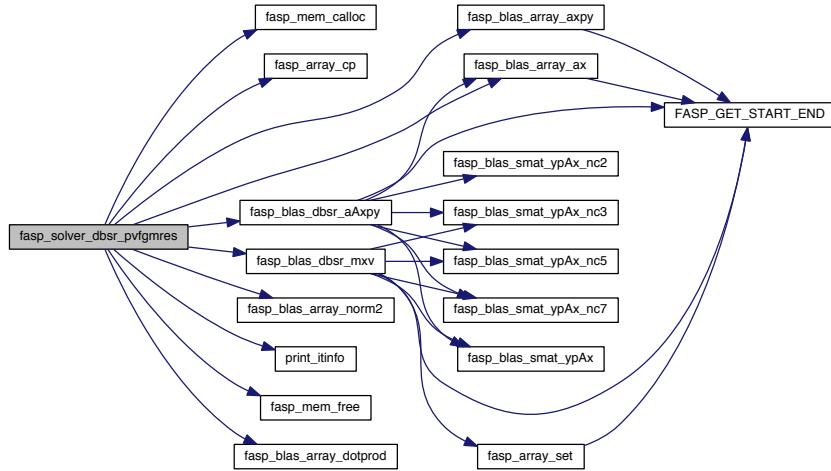
Date

02/05/2012

Modified by Chensong Zhang on 05/01/2012

Definition at line 366 of file pvfgmres.c.

Here is the call graph for this function:



9.73.2.3 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** *print_level*)

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

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>x</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to precond: the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>restart</i>	Restarting steps
<i>stop_type</i>	Stopping criteria type – DOES not support this parameter
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Xiaozhe Hu

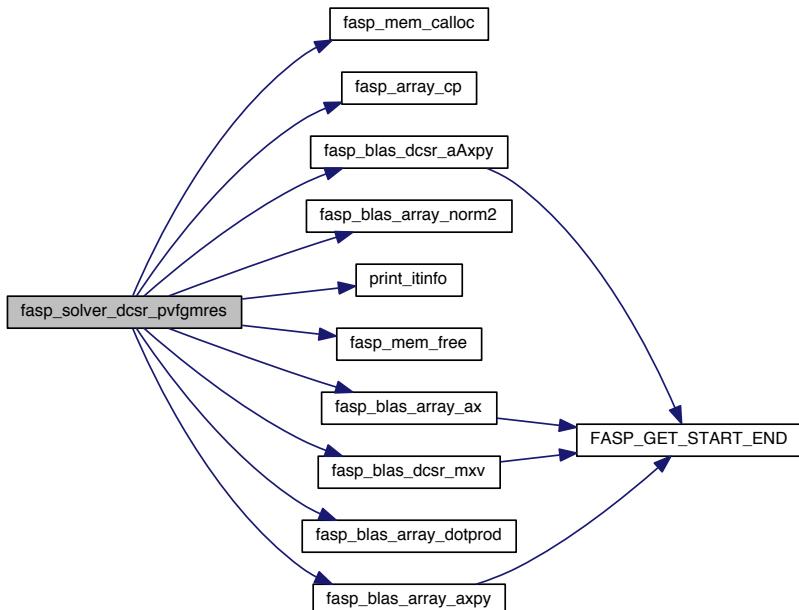
Date

01/04/2012

Modified by Chensong Zhang on 05/01/2012 Modified by Chunsheng Feng on 07/22/2013: Add adaptive memory allocate

Definition at line 54 of file pvfgmres.c.

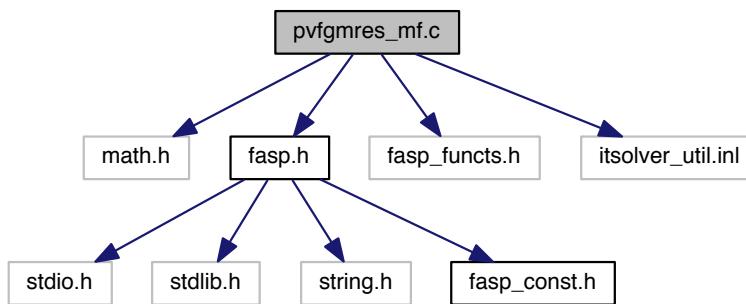
Here is the call graph for this function:



9.74 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"
Include dependency graph for pvfgmres_mf.c:
```



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 print_level)**

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.

9.74.1 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 GMR \leftarrow ES(m) Journal of Computational and Applied Mathematics, 230 (2009) pp. 751-761. UCRL-JRNL-235266.

This file is modified from [pvgmres.c](#)

9.74.2 Function Documentation

- 9.74.2.1 **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 print_level)**

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

<i>mf</i>	Pointer to mxv_matfree : the spmv operation
<i>b</i>	Pointer to dvector: the right hand side
<i>x</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to precond: the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>restart</i>	Restarting steps
<i>stop_type</i>	Stopping criteria type – DOES not support this parameter
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Xiaozhe Hu

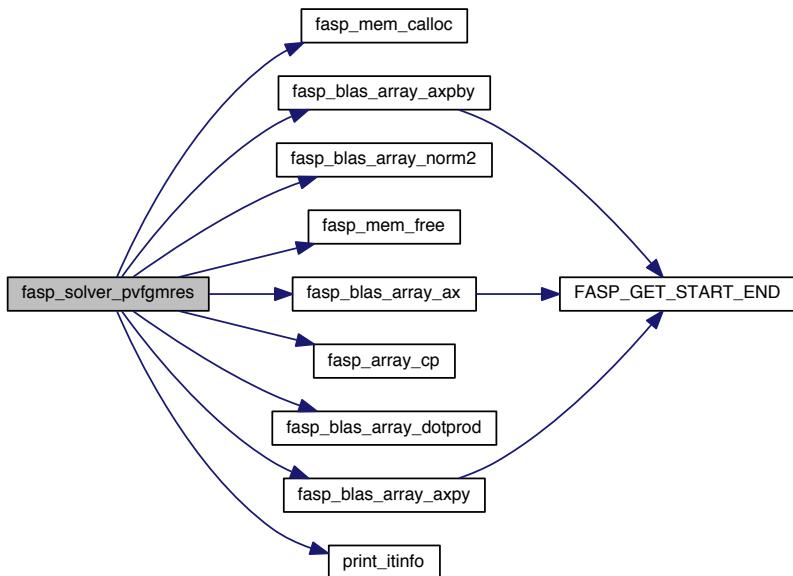
Date

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 56 of file `pvgmres_mf.c`.

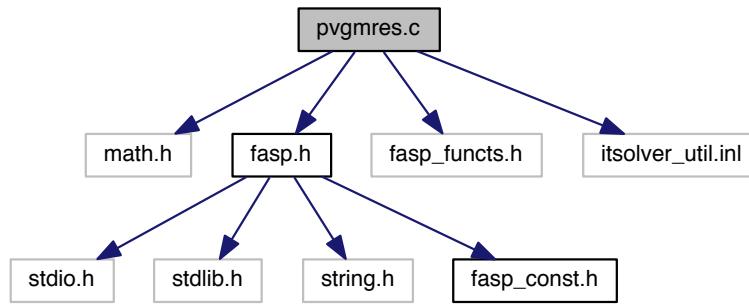
Here is the call graph for this function:



9.75 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"
Include dependency graph for pvgmres.c:
```



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 print_level)`
Right preconditioned GMRES method in which the restart parameter can be adaptively modified during the iteration.
- `INT fasp_solver_bdcsr_pvgmres (block_dCSRmat *A, dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT print_level)`
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 print_level)`
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 print_level)`
Right preconditioned GMRES method in which the restart parameter can be adaptively modified during the iteration.

9.75.1 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 GMRES(m) Journal of Computational and Applied Mathematics, 230 (2009) pp. 751-761. UCRL-JRNL-235266.
 See [spvgmres.c](#) for a safer version

9.75.2 Function Documentation

9.75.2.1 **INT fasp_solver_bdcsr_pvgmres (block_dCSRmat * A, dvector * b, dvector * x, precond * pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT print_level)**

Right preconditioned GMRES method in which the restart parameter can be adaptively modified during the iteration.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector : the right hand side
<i>x</i>	Pointer to dvector : the unknowns
<i>pc</i>	Pointer to precond : the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>restart</i>	Restarting steps
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

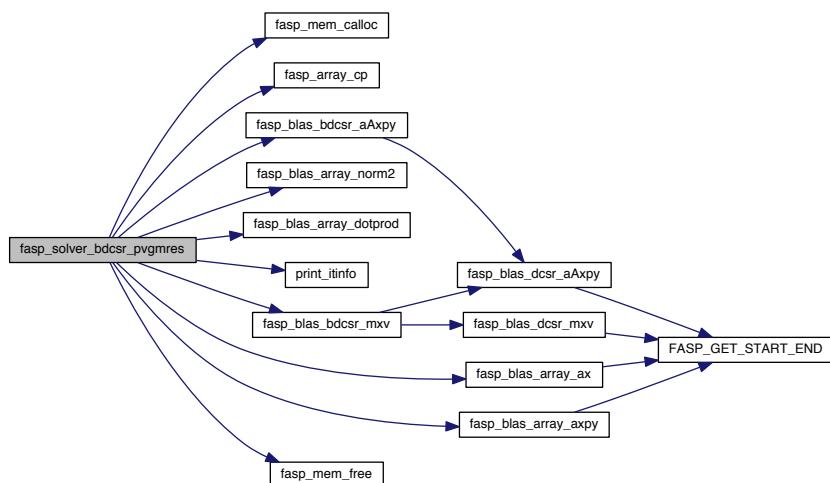
Chensong Zhang

Date

04/05/2013

Definition at line 394 of file pvgmres.c.

Here is the call graph for this function:



```
9.75.2.2 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 print_level )
```

Right preconditioned GMRES method in which the restart parameter can be adaptively modified during the iteration.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>x</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to precond: the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>restart</i>	Restarting steps
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Zhiyang Zhou

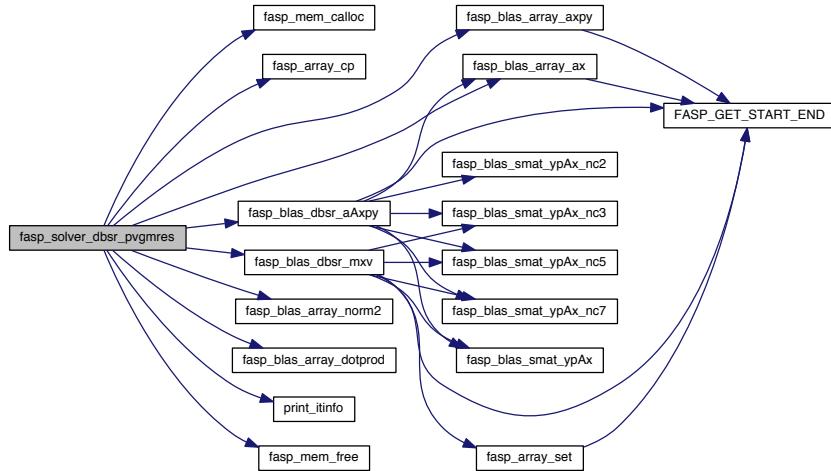
Date

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 739 of file `pvgmres.c`.

Here is the call graph for this function:



9.75.2.3 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** *print_level*)**

Right preconditioned GMRES method in which the restart parameter can be adaptively modified during the iteration.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>x</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to precond: the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>restart</i>	Restarting steps
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Zhiyang Zhou

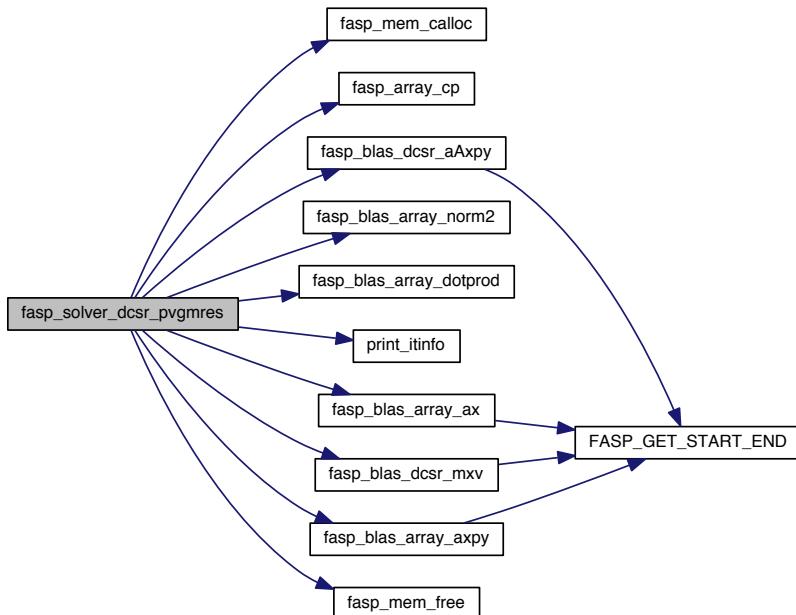
Date

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 52 of file pvgmres.c.

Here is the call graph for this function:



9.75.2.4 **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 print_level*)**

Right preconditioned GMRES method in which the restart parameter can be adaptively modified during the iteration.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to <i>dvector</i> : the right hand side
<i>x</i>	Pointer to <i>dvector</i> : the unknowns
<i>pc</i>	Pointer to <i>precond</i> : the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>restart</i>	Restarting steps
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

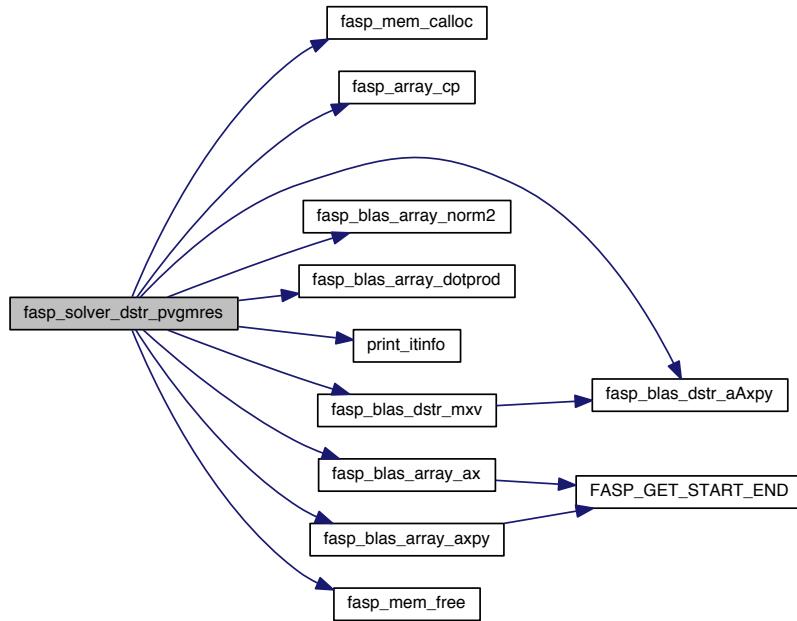
Zhiyang Zhou

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 1084 of file pvgmres.c.

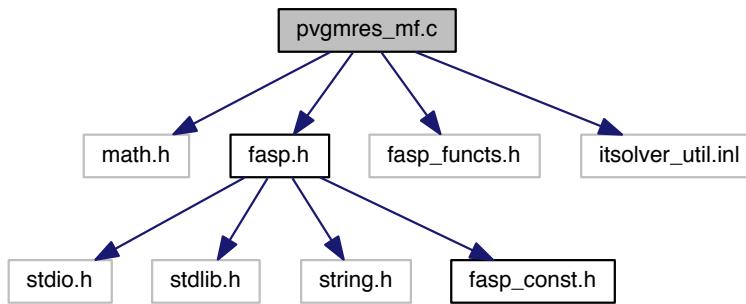
Here is the call graph for this function:



9.76 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"
Include dependency graph for pvgmres_mf.c:
```



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 print_level)**

Solve "Ax=b" using PGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during the iteration.

9.76.1 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 \leftarrow ES(m) Journal of Computational and Applied Mathematics, 230 (2009) pp. 751-761. UCRL-JRNL-235266.

9.76.2 Function Documentation

- 9.76.2.1 **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 *print_level*)**

Solve "Ax=b" using PGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during the iteration.

Parameters

<i>mf</i>	Pointer to mxv_matfree : the spmv operation
<i>b</i>	Pointer to dvector: the right hand side
<i>x</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to precond: the structure of precondition
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>restart</i>	Restarting steps
<i>stop_type</i>	Stopping criteria type – DOES not support this parameter
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

Zhiyang Zhou

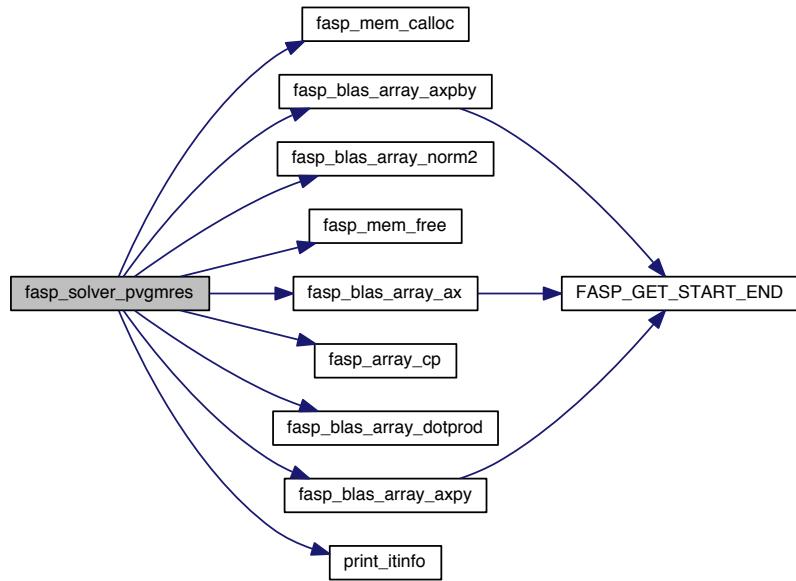
Date

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.

Here is the call graph for this function:

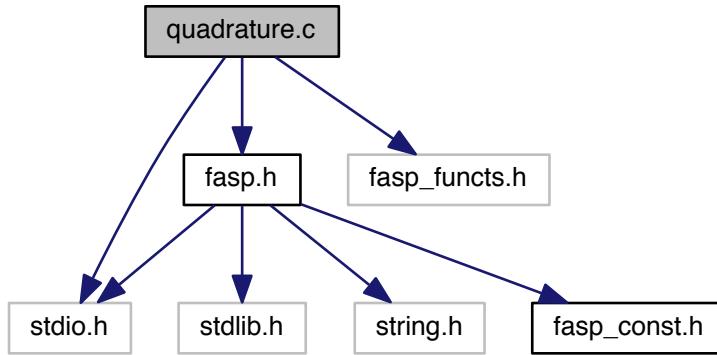


9.77 quadrature.c File Reference

Quadrature rules.

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

Include dependency graph for quadrature.c:



Functions

- void `fasp_quad2d (INT num_qp, INT ncoor, REAL(*quad)[3])`
Initialize Lagrange quadrature points and weights.
- void `fasp_gauss2d (INT num_qp, INT ncoor, REAL(*gauss)[3])`
Initialize Gauss quadrature points and weights.

9.77.1 Detailed Description

Quadrature rules.

9.77.2 Function Documentation

9.77.2.1 void `fasp_gauss2d (INT num_qp, INT ncoor, REAL(*) gauss[3])`

Initialize Gauss quadrature points and weights.

Parameters

<code>num_qp</code>	Number of quadrature points
<code>ncoor</code>	Dimension of space
<code>gauss</code>	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.

Here is the call graph for this function:



9.77.2.2 void fasp_quad2d (INT num_qp, INT ncoor, REAL(*) quad[3])

Initialize Lagrange quadrature points and weights.

Parameters

<code>num_qp</code>	Number of quadrature points
<code>ncoor</code>	Dimension of space
<code>quad</code>	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.

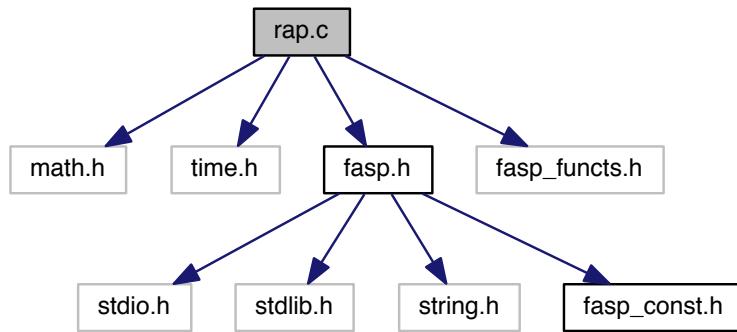
Here is the call graph for this function:



9.78 rap.c File Reference

R*A*P driver.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
Include dependency graph for rap.c:
```



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 *maxrout, INT *ipin, INT *jpin)`

*Compute R*A*P.*

9.78.1 Detailed Description

R*A*P driver.

C-version by Ludmil Zikatanov 2010-04-08

tested 2010-04-08

9.78.2 Function Documentation

- 9.78.2.1 `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 * maxrout, INT * ipin, INT * jpin)`

*Compute R*A*P.*

Author

Ludmil Zikatanov

Date

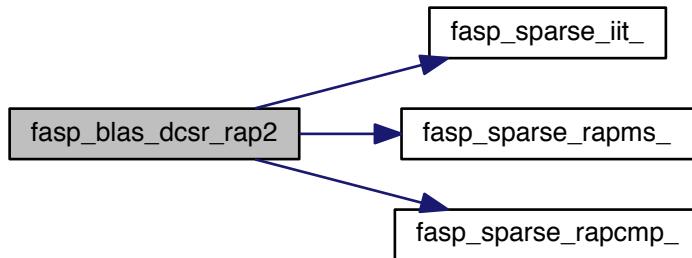
04/08/2010

Note

It uses `dCSRmat` only. The functions called from here are in `sparse_util.c`

Definition at line 33 of file rap.c.

Here is the call graph for this function:



9.79 schwarz.f File Reference

Schwarz smoothers.

Functions/Subroutines

- subroutine **`cut0`** (n, ia, ja, a, iaw, jaw, jblk, iblk, nblk, lwork1, lwork2, lwork3, msize)
- subroutine **`chsize`** (a, b, tol, imin)
- subroutine **`shift`** (nxadj, nadj, n)
- subroutine **`dfs`** (n, ia, ja, nblk, iblk, jblk, lowlink, iedge, numb)
- subroutine **`permat`** (iord, ia, ja, an, n, m, iat, jat, ant)
- subroutine **`pervec`** (iord, u1, u2, n)
- subroutine **`perback`** (iord, u1, u2, n)
- subroutine **`perm0`** (iord, ia, ja, an, n, m, iat, jat, ant)
- subroutine **`icopyv`** (iu, iv, n)
- subroutine **`mxfrm2`** (n, ia, ja, nblk, iblock, jblock, mask, maxa, memt, maxbs)
- subroutine **`sky2ns`** (n, ia, ja, a, nblk, iblock, jblock, mask, maxa, au, al)
- subroutine **`fbgs2ns`** (n, ia, ja, a, x, b, nblk, iblock, jblock, mask, maxa, au, al, rhsloc, memt)
- subroutine **`bbgs2ns`** (n, ia, ja, a, x, b, nblk, iblock, jblock, mask, maxa, au, al, rhsloc, memt)

- subroutine **doluns** (au, al, maxa, nn)
- subroutine **sluns** (au, al, v, maxa, nn)
- subroutine **dolu** (a, maxa, nn)
- subroutine **slvlu** (a, v, maxa, nn)
- subroutine **ijacrs** (ln, ia, ja, a, n, nnz, ir, ic, aij)
- subroutine **sympat** (ln, ia, ja, n, ir, ic, aij)
- subroutine **levels** (inroot, ia, ja, mask, nlvl, iblock, jblock, maxlev)

9.79.1 Detailed Description

Schwarz smoothers.

Author

Ludmil Zikatanov

Date

01/01/2007

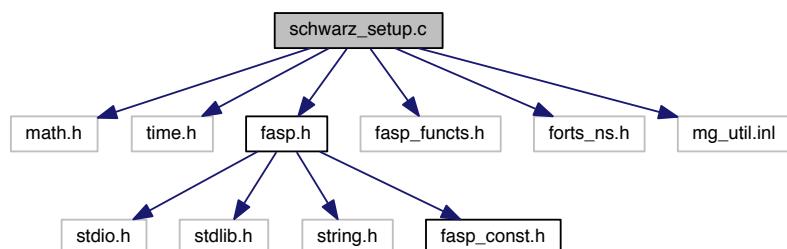
Note

These routines are part of the matching MG method

9.80 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 "forts_ns.h"
#include "mg_util.inl"
Include dependency graph for schwarz_setup.c:
```



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_smoothen` (`Schwarz_data` *`Schwarz`, `Schwarz_param` *`param`, `dvector` *`x`, `dvector` *`b`)
Schwarz smoother: forward sweep.
- void `fasp_dcsr_Schwarz_backward_smoothen` (`Schwarz_data` *`Schwarz`, `Schwarz_param` *`param`, `dvector` *`x`, `dvector` *`b`)
Schwarz smoother: backward sweep.

9.80.1 Detailed Description

Setup phase for the Schwarz methods.

9.80.2 Function Documentation

9.80.2.1 void `fasp_dcsr_Schwarz_backward_smoothen` (`Schwarz_data` * `Schwarz`, `Schwarz_param` * `param`, `dvector` * `x`, `dvector` * `b`)

Schwarz smoother: backward sweep.

Parameters

<code>Schwarz</code>	Pointer to the Schwarz data
<code>param</code>	Pointer to the Schwarz parameter
<code>x</code>	Pointer to solution vector
<code>b</code>	Pointer to right hand

Author

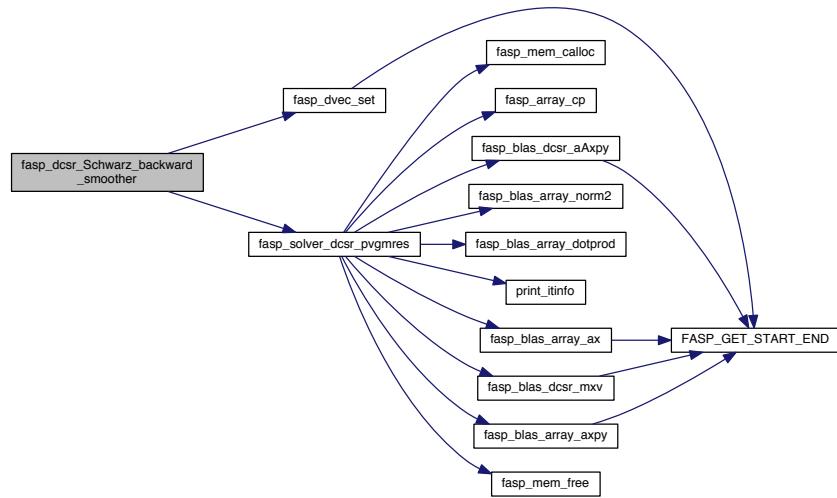
Zheng Li, Chensong Zhang

Date

2014/10/5

Definition at line 404 of file schwarz_setup.c.

Here is the call graph for this function:



9.80.2.2 void fasp_dCSR_Schwarz_forward_smoothe (Schwarz_data * Schwarz, Schwarz_param * param, dvector * x, dvector * b)

Schwarz smoother: forward sweep.

Parameters

<code>Schwarz</code>	Pointer to the Schwarz data
<code>param</code>	Pointer to the Schwarz parameter
<code>x</code>	Pointer to solution vector
<code>b</code>	Pointer to right hand

Author

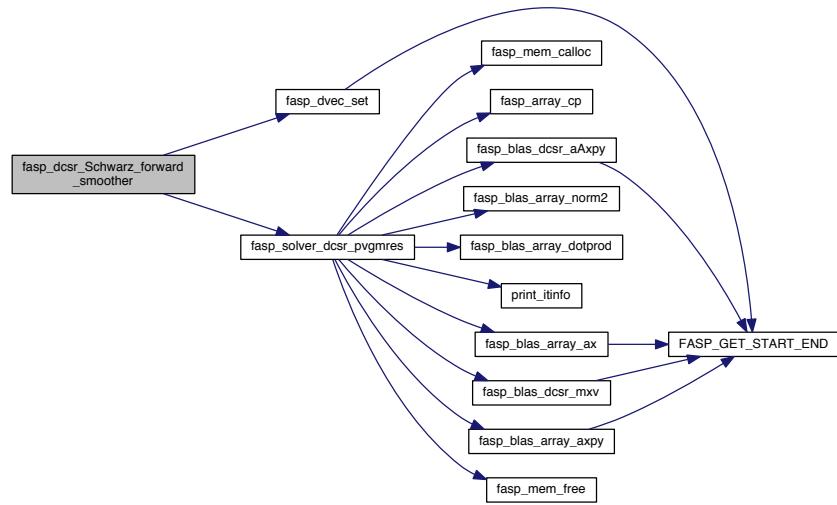
Zheng Li, Chensong Zhang

Date

2014/10/5

Definition at line 294 of file schwarz_setup.c.

Here is the call graph for this function:



9.80.2.3 fasp_Schwarz_get_block_matrix (*Schwarz_data * Schwarz*, *INT nblk*, *INT * iblock*, *INT * jblock*, *INT * mask*)

Form Schwarz partition data.

Parameters

<i>Schwarz</i>	Pointer to the Schwarz data
<i>nblk</i>	Number of partitions
<i>iblock</i>	Pointer to number of vertices on each level
<i>jblock</i>	Pointer to vertices of each level
<i>mask</i>	Pointer to flag array

Author

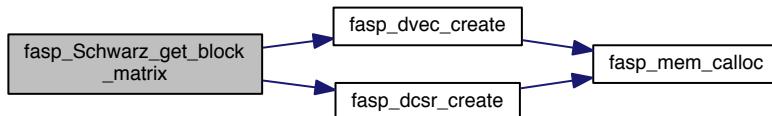
Zheng Li, Chensong Zhang

Date

2014/09/29

Definition at line 35 of file schwarz_setup.c.

Here is the call graph for this function:



9.80.2.4 INT fasp_Schwarz_setup (Schwarz_data * Schwarz, Schwarz_param * param)

Setup phase for the Schwarz methods.

Parameters

<i>Schwarz</i>	Pointer to the Schwarz data
<i>param</i>	Type of the Schwarz method

Returns

FASP_SUCCESS if succeed

Author

Ludmil, Xiaozhe Hu

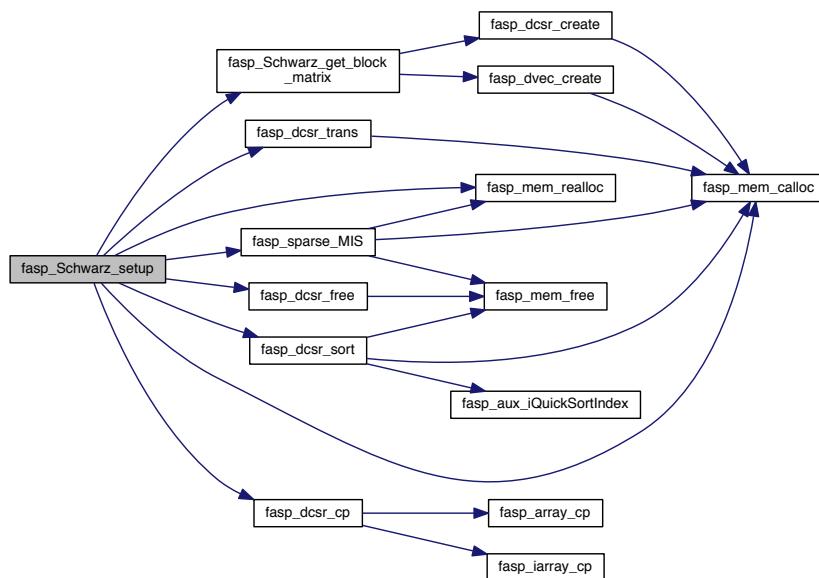
Date

03/22/2011

Modified by Zheng Li on 10/09/2014

Definition at line 126 of file schwarz_setup.c.

Here is the call graph for this function:

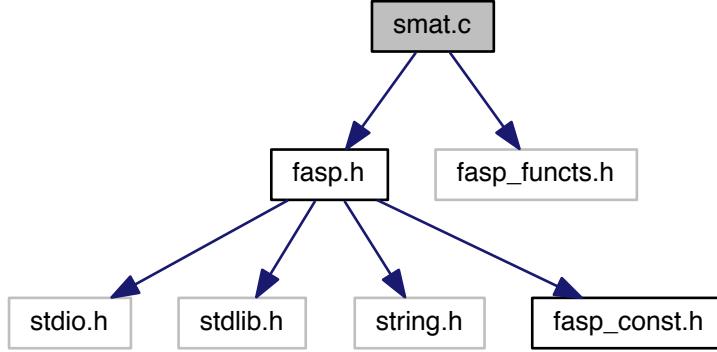


9.81 smat.c File Reference

Simple operations for *small* dense matrices in row-major format.

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

Include dependency graph for smat.c:



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.*
- INT [fasp_blas_smat_inv \(REAL *a, const INT n\)](#)
Compute the inverse matrix of a small full matrix a.
- void [fasp_iden_free \(idenmat *A\)](#)
Free idenmat sparse matrix data memory 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 \(REAL *a, INT n, INT n2\)](#)
*Set a n*n full matrix to be a identity.*
- REAL [fasp_blas_smat_Linfinity \(REAL *A, const INT n\)](#)
Compute the L infinity norm of A.

9.81.1 Detailed Description

Simple operations for *small* dense matrices in row-major format.

9.81.2 Function Documentation

9.81.2.1 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

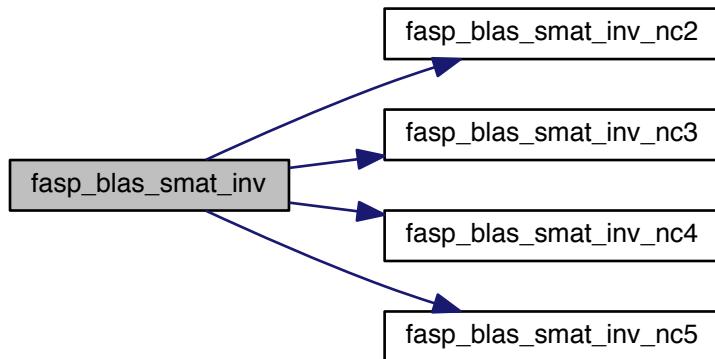
Xiaozhe Hu, Shiquan Zhang

Date

04/21/2010

Definition at line 403 of file smat.c.

Here is the call graph for this function:



9.81.2.2 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 23 of file smat.c.

9.81.2.3 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 59 of file smat.c.

9.81.2.4 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 113 of file smat.c.

9.81.2.5 void fasp blas smat inv nc5 (REAL * a)

Compute the inverse matrix of a 5*5 full matrix A (in place)

Parameters

<code>a</code>	Pointer to the REAL array which stands a 5*5 matrix
----------------	---

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 171 of file smat.c.

9.81.2.6 void fasp_blas_smat_inv_nc7 (REAL * a)

Compute the inverse matrix of a 7*7 matrix a.

Parameters

<code>a</code>	Pointer to the REAL array which stands a 7*7 matrix
----------------	---

Note

This is NOT implemented yet!

Author

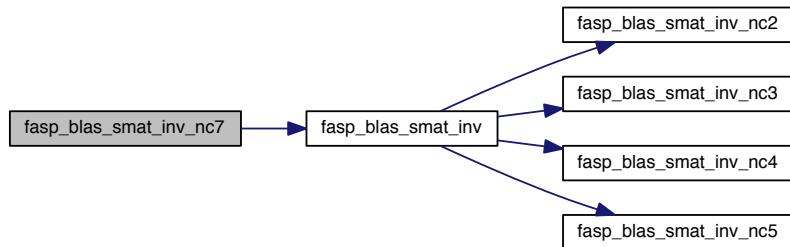
Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 387 of file smat.c.

Here is the call graph for this function:

**9.81.2.7 REAL fasp_blas_smat_Linfinity (REAL * A, const INT n)**

Compute the L infinity norm of A.

Parameters

<i>A</i>	Pointer to the n*n dense matrix
<i>n</i>	the dimension of the dense matrix

Author

Xiaozhe Hu

Date

05/26/2014

Definition at line 673 of file smat.c.

9.81.2.8 void fasp_iden_free (idenmat * A)

Free idenmat sparse matrix data memory space.

Parameters

<i>A</i>	Pointer to the idenmat matrix
----------	-------------------------------

Author

Chensong Zhang

Date

2010/04/03

Definition at line 493 of file smat.c.

Here is the call graph for this function:

**9.81.2.9 void fasp_smat_identity (REAL * a, INT n, INT n2)**

Set a n*n full matrix to be a identity.

Parameters

<i>a</i>	Pointer to the REAL vector which stands for a n*n full matrix
<i>n</i>	Size of full matrix
<i>n2</i>	Length of the REAL vector which stores the n*n full matrix

Author

Xiaozhe Hu

Date

2010/12/25

Definition at line 593 of file smat.c.

9.81.2.10 void fasp_smat_identity_nc2 (REAL * *a*)

Set a 2*2 full matrix to be a identity.

Parameters

<i>a</i>	Pointer to the REAL vector which stands for a 2*2 full matrix
----------	---

Author

Xiaozhe Hu

Date

2011/11/18

Definition at line 513 of file smat.c.

9.81.2.11 void fasp_smat_identity_nc3 (REAL * *a*)

Set a 3*3 full matrix to be a identity.

Parameters

<i>a</i>	Pointer to the REAL vector which stands for a 3*3 full matrix
----------	---

Author

Xiaozhe Hu

Date

2010/12/25

Definition at line 530 of file smat.c.

9.81.2.12 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 547 of file smat.c.

9.81.2.13 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

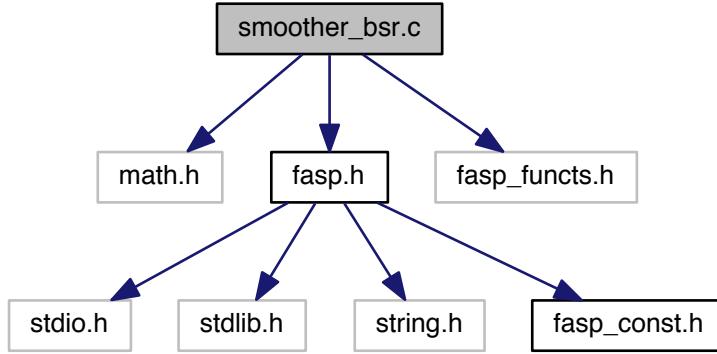
Definition at line 568 of file smat.c.

9.82 smoother_bsr.c File Reference

Smoothers for [dBSRmat](#) matrices.

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

Include dependency graph for smoother_bsr.c:



Functions

- void `fasp_smoothen_dbsr_jacobi (dBSRmat *A, dvector *b, dvector *u)`
Jacobi relaxation.
- void `fasp_smoothen_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_smoothen_dbsr_jacobi1 (dBSRmat *A, dvector *b, dvector *u, REAL *diaginv)`
Jacobi relaxation.
- void `fasp_smoothen_dbsr_gs (dBSRmat *A, dvector *b, dvector *u, INT order, INT *mark)`
Gauss-Seidel relaxation.
- void `fasp_smoothen_dbsr_gs1 (dBSRmat *A, dvector *b, dvector *u, INT order, INT *mark, REAL *diaginv)`
Gauss-Seidel relaxation.
- void `fasp_smoothen_dbsr_gs_ascend (dBSRmat *A, dvector *b, dvector *u, REAL *diaginv)`
Gauss-Seidel relaxation in the ascending order.
- void `fasp_smoothen_dbsr_gs_ascend1 (dBSRmat *A, dvector *b, dvector *u)`
Gauss-Seidel relaxation in the ascending order.
- void `fasp_smoothen_dbsr_gs_descend (dBSRmat *A, dvector *b, dvector *u, REAL *diaginv)`
Gauss-Seidel relaxation in the descending order.
- void `fasp_smoothen_dbsr_gs_descend1 (dBSRmat *A, dvector *b, dvector *u)`
Gauss-Seidel relaxation in the descending order.
- void `fasp_smoothen_dbsr_gs_order1 (dBSRmat *A, dvector *b, dvector *u, REAL *diaginv, INT *mark)`
Gauss-Seidel relaxation in the user-defined order.
- void `fasp_smoothen_dbsr_gs_order2 (dBSRmat *A, dvector *b, dvector *u, INT *mark, REAL *work)`
Gauss-Seidel relaxation in the user-defined order.
- void `fasp_smoothen_dbsr_sor (dBSRmat *A, dvector *b, dvector *u, INT order, INT *mark, REAL weight)`
SOR relaxation.
- void `fasp_smoothen_dbsr_sor1 (dBSRmat *A, dvector *b, dvector *u, INT order, INT *mark, REAL *diaginv, REAL weight)`

SOR relaxation.

- void `fasp_smoothen_dbsr_sor_ascend (dBSRmat *A, dvector *b, dvector *u, REAL *diaginv, REAL weight)`

SOR relaxation in the ascending order.
- void `fasp_smoothen_dbsr_sor_descend (dBSRmat *A, dvector *b, dvector *u, REAL *diaginv, REAL weight)`

SOR relaxation in the descending order.
- void `fasp_smoothen_dbsr_sor_order (dBSRmat *A, dvector *b, dvector *u, REAL *diaginv, INT *mark, REAL weight)`

SOR relaxation in the user-defined order.
- void `fasp_smoothen_dbsr_ilu (dBSRmat *A, dvector *b, dvector *x, void *data)`

ILU method as the smoother in solving $Au=b$ with multigrid method.

9.82.1 Detailed Description

Smoothers for `dBSRmat` matrices.

9.82.2 Function Documentation

9.82.2.1 void `fasp_smoothen_dbsr_gs (dBSRmat * A, dvector * b, dvector * u, INT order, INT * mark)`

Gauss-Seidel relaxation.

Parameters

<i>A</i>	Pointer to <code>dBSRmat</code> : the coefficient matrix
<i>b</i>	Pointer to <code>dvector</code> : the right hand side
<i>u</i>	Pointer to <code>dvector</code> : the unknowns (IN: initial, OUT: approximation)
<i>order</i>	Flag to indicate the order for smoothing If <i>mark</i> = NULL ASCEND 12: in ascending order DE← SCEND 21: in descending order If <i>mark</i> != NULL: in the user-defined order
<i>mark</i>	Pointer to NULL or to the user-defined ordering

Author

Zhiyang Zhou

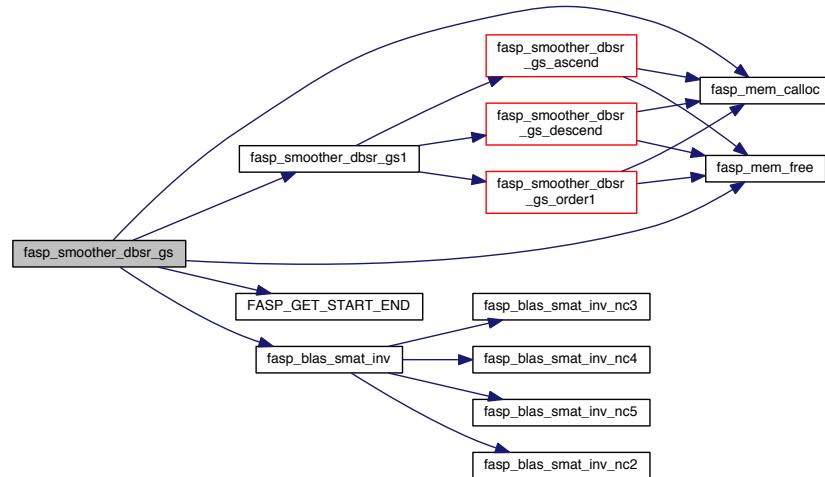
Date

2010/10/25

Modified by Chunsheng Feng, Zheng Li on 08/03/2012

Definition at line 415 of file smoother_bsr.c.

Here is the call graph for this function:

**9.82.2.2 void fasp_smoothen_dbsr_gs1 (dBsrmat * A, dvector * b, dvector * u, INT order, INT * mark, REAL * diaginv)**

Gauss-Seidel relaxation.

Parameters

<i>A</i>	Pointer to dBSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector : the right hand side
<i>u</i>	Pointer to dvector : the unknowns (IN: initial, OUT: approximation)
<i>order</i>	Flag to indicate the order for smoothing If mark = NULL ASCEND 12: in ascending order DE← SCEND 21: in descending order If mark != NULL: in the user-defined order
<i>mark</i>	Pointer to NULL or to the user-defined ordering
<i>diaginv</i>	Inverses for all the diagonal blocks of A

Author

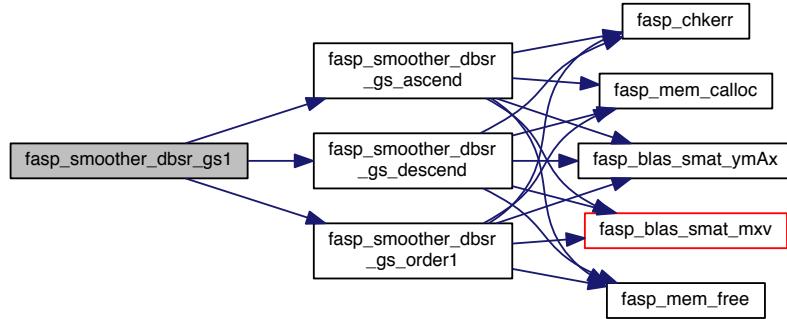
Zhiyang Zhou

Date

2010/10/25

Definition at line 535 of file smoother_bsr.c.

Here is the call graph for this function:



9.82.2.3 void fasp_smoothen_dbsr_gs_ascend (dBsrmat * A, dvector * b, dvector * u, REAL * diaginv)

Gauss-Seidel relaxation in the ascending order.

Parameters

<i>A</i>	Pointer to dBsrmat : the coefficient matrix
<i>b</i>	Pointer to dvector : the right hand side
<i>u</i>	Pointer to dvector : the unknowns (IN: initial guess, OUT: approximation)
<i>diaginv</i>	Inverses for all the diagonal blocks of A

Author

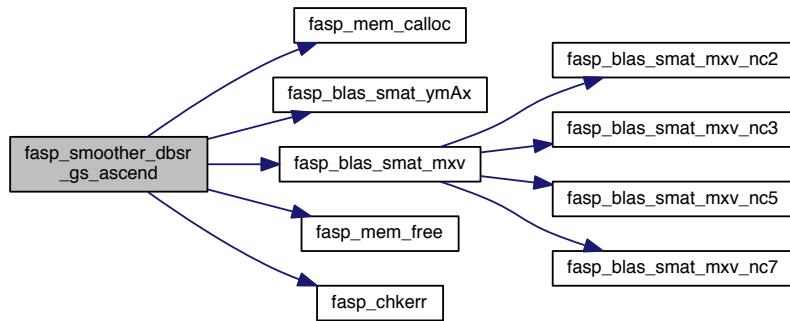
Zhiyang Zhou

Date

2010/10/25

Definition at line 572 of file smoother_bsr.c.

Here is the call graph for this function:

9.82.2.4 void fasp_smoothen_dbsr_gs_ascend1 (**dBSRmat** * *A*, **dvector** * *b*, **dvector** * *u*)

Gauss-Seidel relaxation in the ascending order.

Parameters

<i>A</i>	Pointer to dBSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector : the right hand side
<i>u</i>	Pointer to dvector : the unknowns (IN: initial guess, OUT: approximation)

Author

Xiaozhe

Date

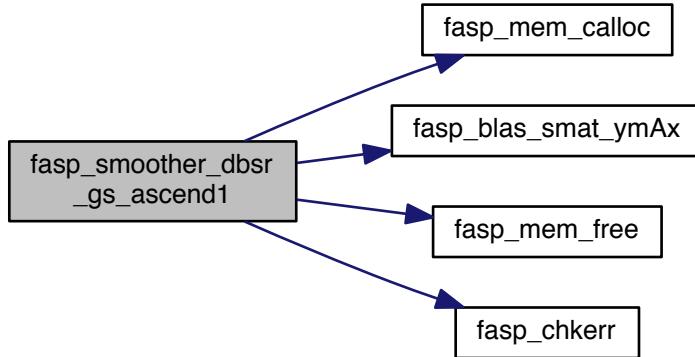
01/01/2014

Note

The only difference between the functions 'fasp_smoothen_dbsr_gs_ascend1' and 'fasp_smoothen_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 645 of file smoother_bsr.c.

Here is the call graph for this function:



9.82.2.5 void fasp_smoothen_dbsr_gs_descend (**dBSRmat** * *A*, **dvector** * *b*, **dvector** * *u*, **REAL** * *diaginv*)

Gauss-Seidel relaxation in the descending order.

Parameters

<i>A</i>	Pointer to dBSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector : the right hand side
<i>u</i>	Pointer to dvector : the unknowns (IN: initial guess, OUT: approximation)
<i>diaginv</i>	Inverses for all the diagonal blocks of <i>A</i>

Author

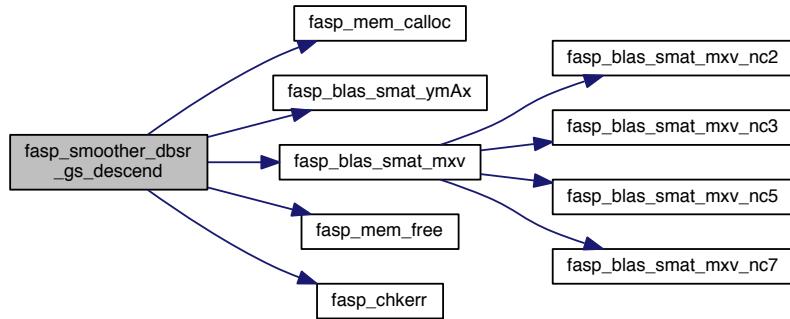
Zhiyang Zhou

Date

2010/10/25

Definition at line 716 of file smoother_bsr.c.

Here is the call graph for this function:



9.82.2.6 void fasp_smoothen_dbsr_gs_descend1 (dBsrmat * A, dvector * b, dvector * u)

Gauss-Seidel relaxation in the descending order.

Parameters

<i>A</i>	Pointer to dBsrmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns (IN: initial guess, OUT: approximation)

Author

Xiaozhe Hu

Date

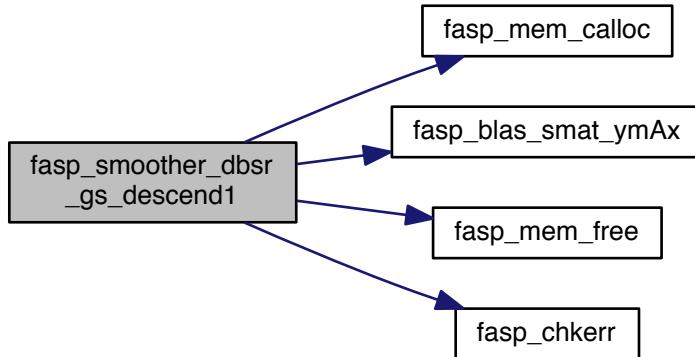
01/01/2014

Note

The only difference between the functions 'fasp_smoothen_dbsr_gs_ascend1' and 'fasp_smoothen_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 790 of file smoother_bsr.c.

Here is the call graph for this function:



9.82.2.7 void fasp_smoothen_dbsr_gs_order1 (dBsrmat * A, dvector * b, dvector * u, REAL * diaginv, INT * mark)

Gauss-Seidel relaxation in the user-defined order.

Parameters

<i>A</i>	Pointer to dBsrmat : the coefficient matrix
<i>b</i>	Pointer to dvector : the right hand side
<i>u</i>	Pointer to dvector : the unknowns (IN: initial guess, OUT: approximation)
<i>diaginv</i>	Inverses for all the diagonal blocks of A
<i>mark</i>	Pointer to the user-defined ordering

Author

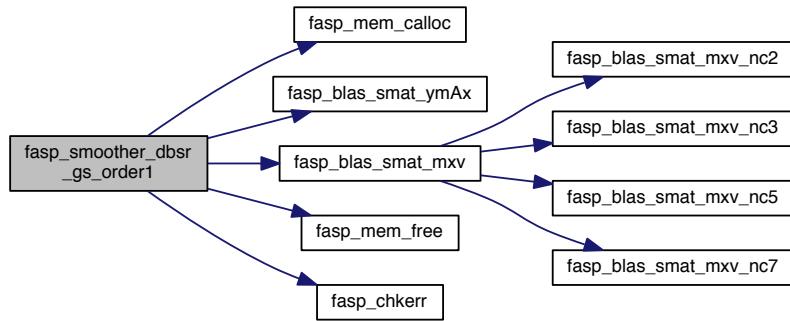
Zhiyang Zhou

Date

2010/10/25

Definition at line 862 of file smoother_bsr.c.

Here is the call graph for this function:

**9.82.2.8 void fasp_smoothen_dbsr_gs_order2 (dBsrmat * A, dvector * b, dvector * u, INT * mark, REAL * work)**

Gauss-Seidel relaxation in the user-defined order.

Parameters

<i>A</i>	Pointer to dBsrmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns (IN: initial guess, OUT: approximation)
<i>mark</i>	Pointer to the user-defined ordering
<i>work</i>	Work temp array

Author

Zhiyang Zhou

Date

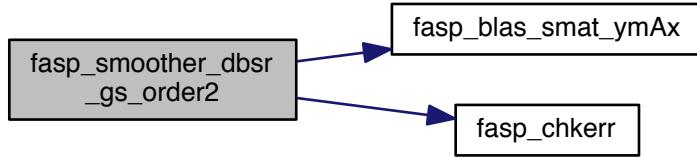
2010/11/08

Note

The only difference between the functions 'fasp_smoothen_dbsr_gs_order2' and 'fasp_smoothen_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 940 of file smoother_bsr.c.

Here is the call graph for this function:



9.82.2.9 void fasp_smoothen_dbsr_ilu (dBsrmat * A, dvector * b, dvector * x, void * data)

ILU method as the smoother in solving $Au=b$ with multigrid method.

Parameters

<i>A</i>	Pointer to dBsrmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>x</i>	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
<i>data</i>	Pointer to user defined data

Author

Zhiyang Zhou

Date

2010/10/25

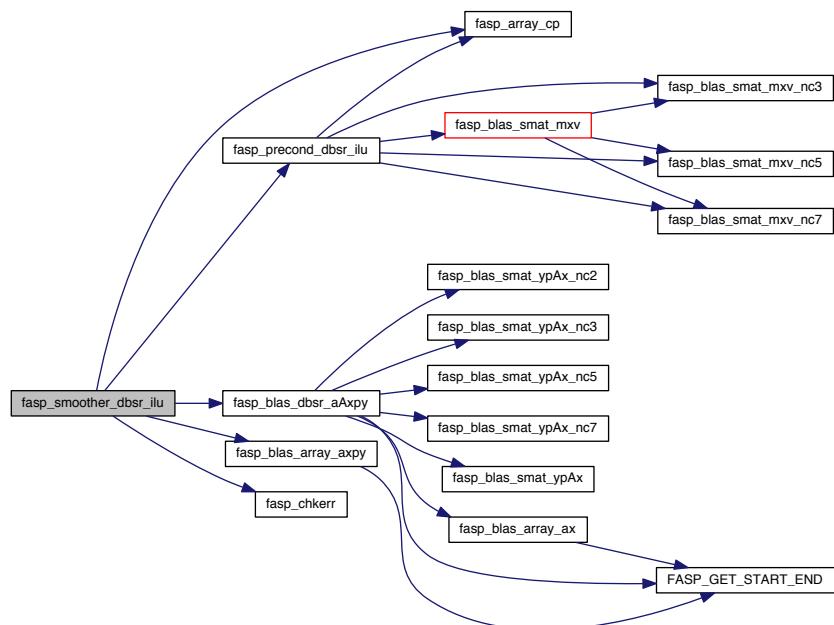
form residual $z_r = b - Ax$

solve LU z=zr

$x=x+z$

Definition at line 1573 of file smoother_bsr.c.

Here is the call graph for this function:



9.82.2.10 void fasp_smoothen_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

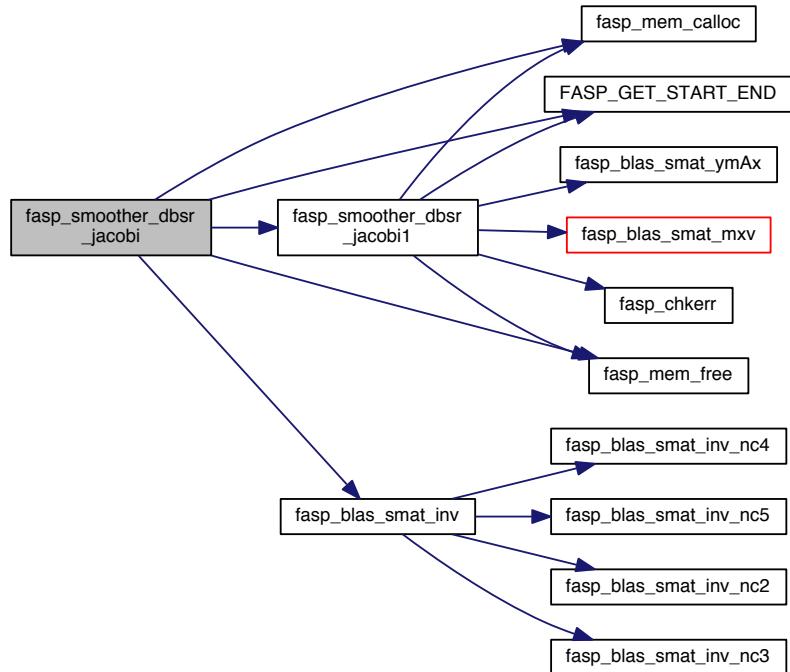
Date

2010/10/25

Modified by Chunsheng Feng, Zheng Li on 08/02/2012

Definition at line 35 of file smoother_bsr.c.

Here is the call graph for this function:

9.82.2.11 void fasp_smoothen_dbsr_jacobi1 (**dBSRmat** * *A*, **dvector** * *b*, **dvector** * *u*, **REAL** * *diaginv*)

Jacobi relaxation.

Parameters

<i>A</i>	Pointer to dBSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector : the right hand side
<i>u</i>	Pointer to dvector : the unknowns (IN: initial, OUT: approximation)
<i>diaginv</i>	Inverses for all the diagonal blocks of <i>A</i>

Author

Zhiyang Zhou

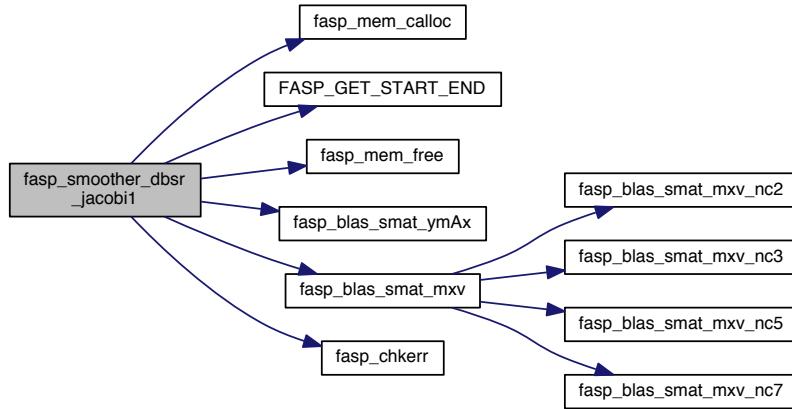
Date

2010/10/25

Modified by Chunsheng Feng, Zheng Li on 08/03/2012

Definition at line 259 of file smoother_bsr.c.

Here is the call graph for this function:

**9.82.2.12 void fasp_smoothen_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

<i>A</i>	Pointer to dBSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector : the right hand side
<i>u</i>	Pointer to dvector : the unknowns (IN: initial, OUT: approximation)
<i>diaginv</i>	Inverse of the diagonal entries

Author

Zhiyang Zhou

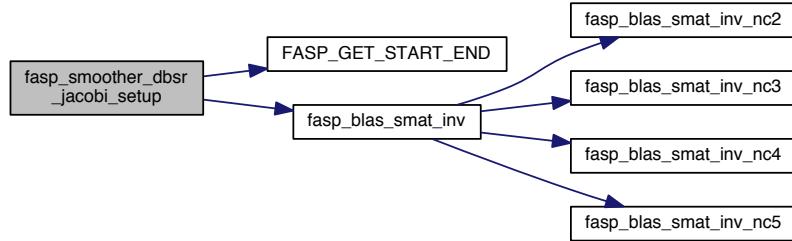
Date

10/25/2010

Modified by Chunsheng Feng, Zheng Li on 08/02/2012

Definition at line 150 of file smoother_bsr.c.

Here is the call graph for this function:



9.82.2.13 void fasp_smoothen_dbsr_sor(dBsrMat * A, dvector * b, dvector * u, INT order, INT * mark, REAL weight)

SOR relaxation.

Parameters

<i>A</i>	Pointer to <code>dBSRmat</code> : the coefficient matrix
<i>b</i>	Pointer to <code>dvector</code> : the right hand side
<i>u</i>	Pointer to <code>dvector</code> : the unknowns (IN: initial guess, OUT: approximation)
<i>order</i>	Flag to indicate the order for smoothing If <i>mark</i> = NULL ASCEND 12: in ascending order DE \leftarrow SCEND 21: in descending order If <i>mark</i> != NULL: in the user-defined order
<i>mark</i>	Pointer to NULL or to the user-defined ordering
<i>weight</i>	Over-relaxation weight

Author

Zhiyang Zhou

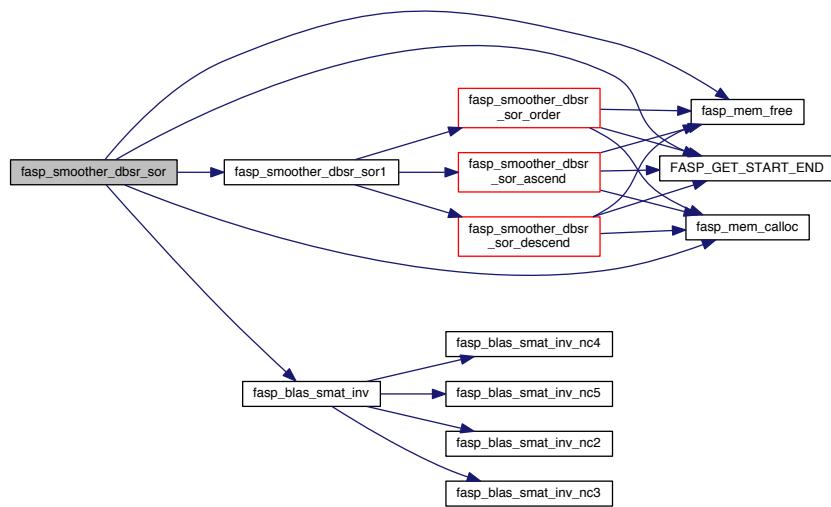
Date

2010/10/25

Modified by Chunsheng Feng, Zheng Li on 08/03/2012

Definition at line 1019 of file smoother_bsr.c.

Here is the call graph for this function:



9.82.2.14 void fasp_smoothen_dbsr_sor1 (**dBSRmat * *A*, **dvector** * *b*, **dvector** * *u*, INT *order*, INT * *mark*, REAL * *diaginv*, REAL *weight*)**

SOR relaxation.

Parameters

<i>A</i>	Pointer to dBSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector : the right hand side
<i>u</i>	Pointer to dvector : the unknowns (IN: initial guess, OUT: approximation)
<i>order</i>	Flag to indicate the order for smoothing If mark = NULL ASCEND 12: in ascending order DE \leftarrow SCEND 21: in descending order If mark != NULL: in the user-defined order
<i>mark</i>	Pointer to NULL or to the user-defined ordering
<i>diaginv</i>	Inverses for all the diagonal blocks of <i>A</i>
<i>weight</i>	Over-relaxation weight

Author

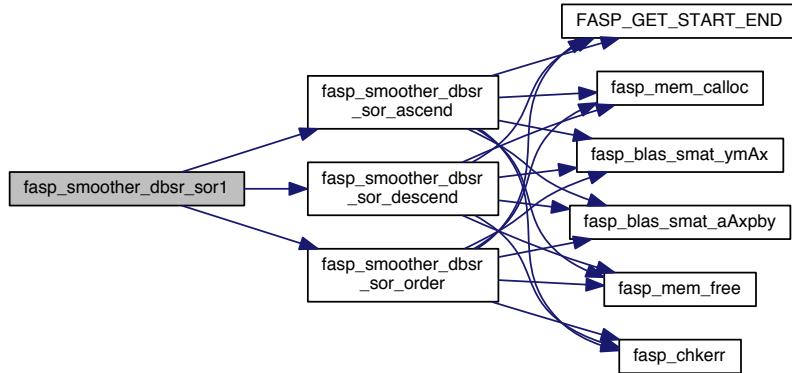
Zhiyang Zhou

Date

2010/10/25

Definition at line 1141 of file smoother_bsr.c.

Here is the call graph for this function:



9.82.2.15 void fasp_smoothen_dbsr_sor_ascend (dBsrmat * A, dvector * b, dvector * u, REAL * diaginv, REAL weight)

SOR relaxation in the ascending order.

Parameters

<i>A</i>	Pointer to <code>dBSRmat</code> : the coefficient matrix
<i>b</i>	Pointer to <code>dvector</code> : the right hand side
<i>u</i>	Pointer to <code>dvector</code> : the unknowns (IN: initial guess, OUT: approximation)
<i>diaginv</i>	Inverses for all the diagonal blocks of <i>A</i>
<i>weight</i>	Over-relaxation weight

Author

Zhiyang Zhou

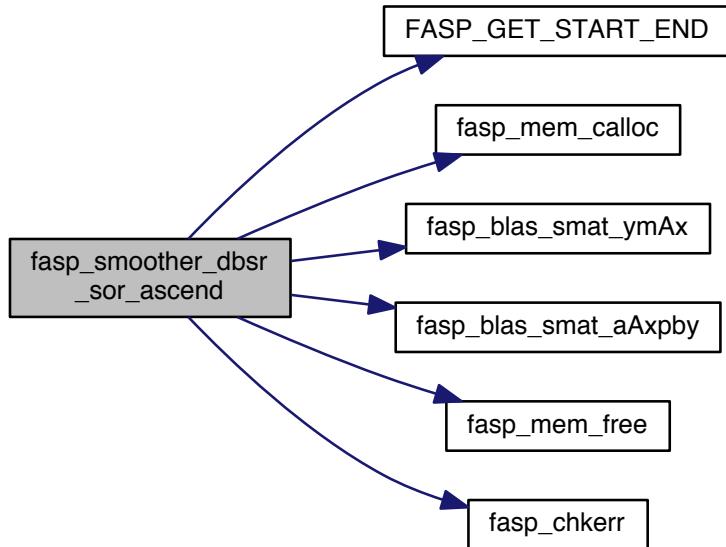
Date

2010/10/25

Modified by Chunsheng Feng, Zheng Li on 2012/09/04

Definition at line 1182 of file smoother_bsr.c.

Here is the call graph for this function:



9.82.2.16 void fasp_smoothen_dbsr_sor_descend (**dBSRmat** * *A*, **dvector** * *b*, **dvector** * *u*, **REAL** * *diaginv*, **REAL** *weight*)

SOR relaxation in the descending order.

Parameters

<i>A</i>	Pointer to dBSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector : the right hand side
<i>u</i>	Pointer to dvector : the unknowns (IN: initial guess, OUT: approximation)
<i>diaginv</i>	Inverses for all the diagonal blocks of <i>A</i>
<i>weight</i>	Over-relaxation weight

Author

Zhiyang Zhou

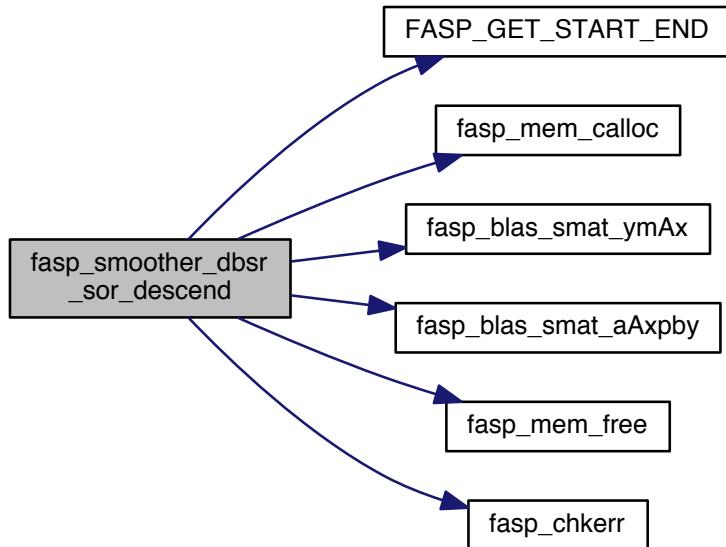
Date

2010/10/25

Modified by Chunsheng Feng, Zheng Li on 2012/09/04

Definition at line 1311 of file smoother_bsr.c.

Here is the call graph for this function:



9.82.2.17 void fasp_smoothen_dbsr_sor_order (dBsrmat * A, dvector * b, dvector * u, REAL * diaginv, INT * mark, REAL weight)

SOR relaxation in the user-defined order.

Parameters

<i>A</i>	Pointer to dBsrmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
<i>diaginv</i>	Inverses for all the diagonal blocks of A
<i>mark</i>	Pointer to the user-defined ordering
<i>weight</i>	Over-relaxation weight

Author

Zhiyang Zhou

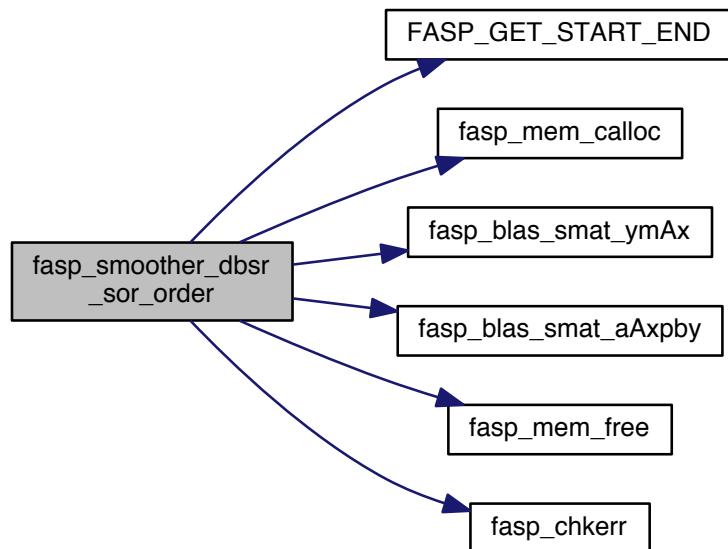
Date

2010/10/25

Modified by Chunsheng Feng, Zheng Li on 2012/09/04

Definition at line 1442 of file smoother_bsr.c.

Here is the call graph for this function:

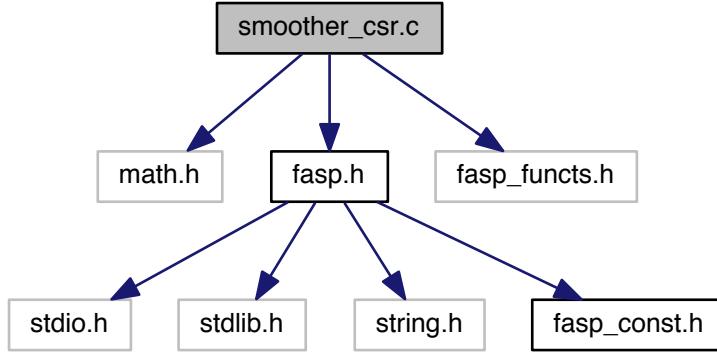


9.83 smoother_csr.c File Reference

Smoothers for [dCSRmat](#) matrices.

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

Include dependency graph for smoother_csr.c:



Functions

- void [fasp_smoothen_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_smoothen_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_smoothen_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_smoothen_dcsr_sgs](#) (dvector *u, dCSRmat *A, dvector *b, INT L)

Symmetric Gauss-Seidel method as a smoother.
- void [fasp_smoothen_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_smoothen_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_smoothen_dcsr_ilu](#) (dCSRmat *A, dvector *b, dvector *x, void *data)

ILU method as a smoother.
- void [fasp_smoothen_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_smoothen_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_smoothen_dcsr_gs_rb3d](#) (dvector *u, dCSRmat *A, dvector *b, INT L, INT order, INT *mark, INT maximap, INT nx, INT ny, INT nz)

Colored Gauss-Seidel smoother for Au=b.

9.83.1 Detailed Description

Smoothers for `dCSRmat` matrices.

9.83.2 Function Documentation

9.83.2.1 `void fasp_smoothe_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

<i>u</i>	Pointer to <code>dvector</code> : the unknowns (IN: initial, OUT: approximation)
<i>i_1</i>	Starting index
<i>i_n</i>	Ending index
<i>s</i>	Increasing step
<i>A</i>	Pointer to <code>dCSRmat</code> : the coefficient matrix
<i>b</i>	Pointer to <code>dvector</code> : the right hand side
<i>L</i>	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`.

Here is the call graph for this function:



9.83.2.2 `void fasp_smoothe_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

<i>u</i>	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
<i>A</i>	Pointer to dBSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>L</i>	Number of iterations
<i>mark</i>	C/F marker array
<i>order</i>	C/F ordering: -1: F-first; 1: C-first

Author

Zhiyang Zhou

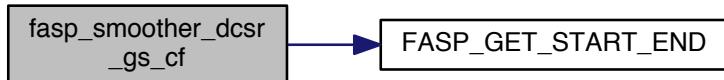
Date

11/12/2010

Modified by Chunsheng Feng, Xiaoqiang Yue on 05/24/2012

Definition at line 364 of file smoother_csr.c.

Here is the call graph for this function:



9.83.2.3 `void fasp_smoothen_dcsr_gs_rb3d(dvector * u, dCSRmat * A, dvector * b, INT L, INT order, INT * mark, INT maximap, INT nx, INT ny, INT nz)`

Colored Gauss-Seidel smoother for Au=b.

Parameters

<i>u</i>	Initial guess and the new approximation to the solution
<i>A</i>	Pointer to stiffness matrix
<i>b</i>	Pointer to right hand side
<i>L</i>	Number of iterations
<i>order</i>	Ordering: -1: Forward; 1: Backward
<i>mark</i>	Marker for C/F points
<i>maximap</i>	Size of IMAP

<i>nx</i>	Number vertex of X direction
<i>ny</i>	Number vertex of Y direction
<i>nz</i>	Number vertex of Z direction

Author

Chunsheng Feng

Date

02/08/2012

Definition at line 1426 of file smoother_csr.c.

9.83.2.4 `void fasp_smoothen_dcsr_ilu (dCSRmat * A, dvector * b, dvector * x, void * data)`

ILU method as a smoother.

Parameters

<i>A</i>	Pointer to <code>dBSRmat</code> : the coefficient matrix
<i>b</i>	Pointer to <code>dvector</code> : the right hand side
<i>x</i>	Pointer to <code>dvector</code> : the unknowns (IN: initial, OUT: approximation)
<i>data</i>	Pointer to user defined data

Author

Shiquan Zhang, Xiaozhe Hu

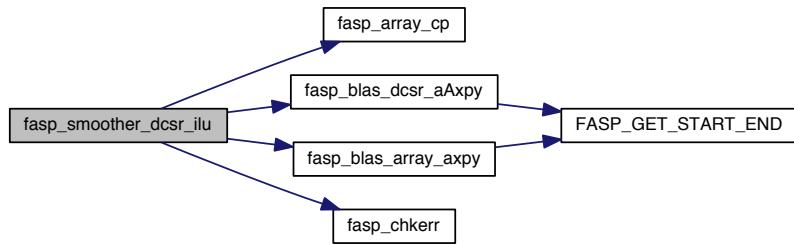
Date

2010/11/12

form residual $zr = b - Ax$

Definition at line 1067 of file smoother_csr.c.

Here is the call graph for this function:



```
9.83.2.5 void fasp_smoothen_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

<i>u</i>	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
<i>i_1</i>	Starting index
<i>i_n</i>	Ending index
<i>s</i>	Increasing step
<i>A</i>	Pointer to dBSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>L</i>	Number of iterations

Author

Xuehai Huang, Chensong Zhang

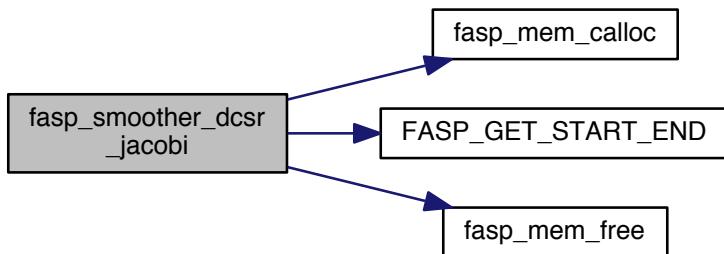
Date

09/26/2009

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

Definition at line 59 of file smoother_csr.c.

Here is the call graph for this function:



9.83.2.6 void fasp_smoothen_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

<i>u</i>	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
<i>i_1</i>	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

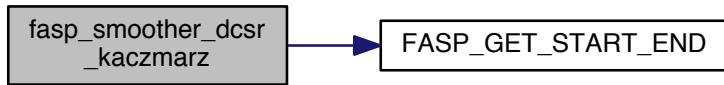
Date

2010/11/12

Modified by Chunsheng Feng, Zheng Li on 2012/09/01

Definition at line 1145 of file smoother_csr.c.

Here is the call graph for this function:



9.83.2.7 void fasp_smoothen_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

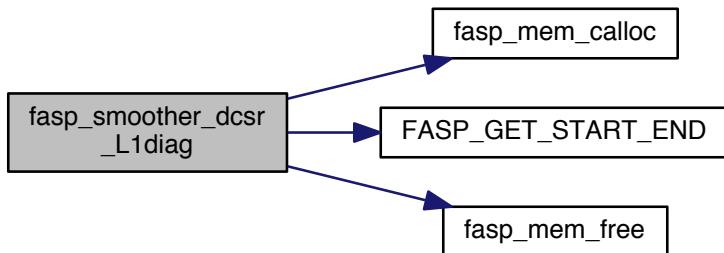
Date

01/26/2011

Modified by Chunsheng Feng, Zheng Li on 09/01/2012

Definition at line 1286 of file smoother_csr.c.

Here is the call graph for this function:

**9.83.2.8 void fasp_smoothen_dcsr_sgs (dvector * u, dCSRmat * A, dvector * b, INT L)**

Symmetric Gauss-Seidel method as a smoother.

Parameters

<i>u</i>	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>L</i>	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.

Here is the call graph for this function:



9.83.2.9 `void fasp_smoothen_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

<i>u</i>	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
<i>i_1</i>	Starting index
<i>i_n</i>	Ending index
<i>s</i>	Increasing step
<i>A</i>	Pointer to <code>dCSRmat</code> : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>L</i>	Number of iterations
<i>w</i>	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.

Here is the call graph for this function:

**9.83.2.10** `void fasp_smoothen_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**

<i>u</i>	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
<i>A</i>	Pointer to <code>dCSRmat</code> : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>L</i>	Number of iterations
<i>w</i>	Over-relaxation weight
<i>mark</i>	C/F marker array
<i>order</i>	C/F ordering: -1: F-first; 1: C-first

Author

Zhiyang Zhou

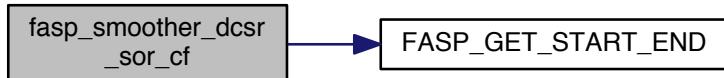
Date

2010/11/12

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

Definition at line 873 of file smoother_csr.c.

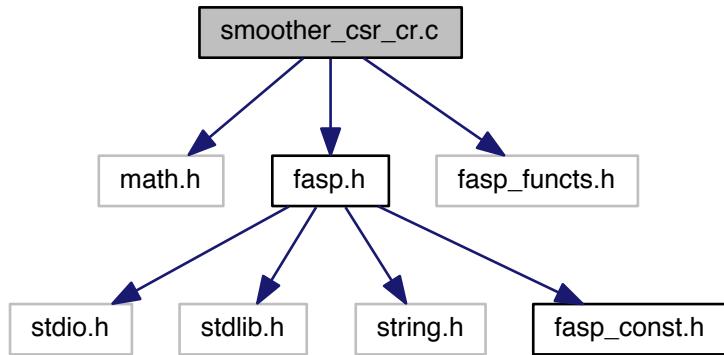
Here is the call graph for this function:



9.84 smoother_csr_cr.c File Reference

Smoothers for dCSRmat matrices using compatible relaxation.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
Include dependency graph for smoother_csr_cr.c:
```



Functions

- void `fasp_smoothen_dcsr_gscr (INT pt, INT n, REAL *u, INT *ia, INT *ja, REAL *a, REAL *b, INT L, INT *CF)`
Gauss Seidel method restricted to a block.

9.84.1 Detailed Description

Smoothers for dCSRmat matrices using compatible relaxation.

Note

Restricted-smoothers for compatible relaxation, C/F smoothing, etc.

9.84.2 Function Documentation

9.84.2.1 `void fasp_smoothen_dcsr_gscr (INT pt, INT n, REAL * u, INT * ia, INT * ja, REAL * a, REAL * b, INT L, INT * CF)`

Gauss Seidel method restricted to a block.

Parameters

<i>pt</i>	Relax type, e.g., cpt, fpt, etc..
<i>n</i>	Number of variables
<i>u</i>	Iterated solution
<i>ia</i>	Row pointer
<i>ja</i>	Column index
<i>a</i>	Pointers to sparse matrix values in CSR format
<i>b</i>	Pointer to right hand side – remove later also as MG relaxation on error eqn
<i>L</i>	Number of iterations
<i>CF</i>	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.

Here is the call graph for this function:

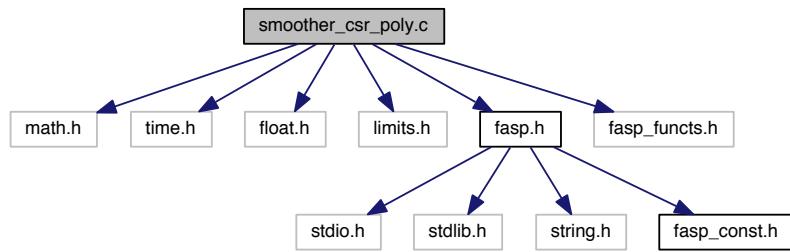


9.85 smoother_csr_poly.c File Reference

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"
```

Include dependency graph for `smoother_csr_poly.c`:



Functions

- `void fasp_smoothen_dcsr_poly (dCSRmat *Amat, dvector *brhs, dvector *usol, INT n, INT ndeg, INT L)`
 $\text{poly approx to } A^{-1} \text{ as MG smoother}$
- `void fasp_smoothen_dcsr_poly_old (dCSRmat *Amat, dvector *brhs, dvector *usol, INT n, INT ndeg, INT L)`
 $\text{poly approx to } A^{-1} \text{ as MG smoother: JK<Z2010}$

9.85.1 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-level methods", 2013.

9.85.2 Function Documentation

9.85.2.1 void `fasp_smoothen_dcsr_poly (dCSRmat * Amat, dvector * brhs, dvector * usol, INT n, INT ndeg, INT L)`

$\text{poly approx to } A^{-1} \text{ as MG smoother}$

Parameters

<i>Amat</i>	Pointer to stiffness matrix, consider square matrix.
<i>brhs</i>	Pointer to right hand side
<i>usol</i>	Pointer to solution
<i>n</i>	Problem size
<i>ndeg</i>	Degree of poly
<i>L</i>	Number of iterations

Author

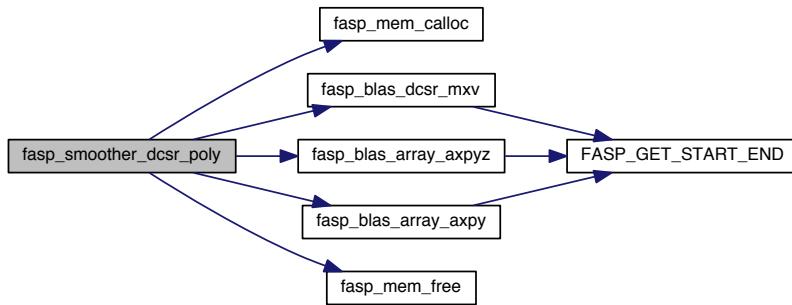
Fei Cao, Xiaozhe Hu

Date

05/24/2012

Definition at line 48 of file smoother_csr_poly.c.

Here is the call graph for this function:



9.85.2.2 void fasp_smoothen_dcsr_poly_old (dCSRmat * Amat, dvector * brhs, dvector * usol, INT n, INT ndeg, INT L)

poly approx to A^{-1} as MG smoother: JK<Z2010

Parameters

<i>Amat</i>	Pointer to stiffness matrix
<i>brhs</i>	Pointer to right hand side
<i>usol</i>	Pointer to solution
<i>n</i>	Problem size
<i>ndeg</i>	Degree of poly
<i>L</i>	Number of iterations

Author

James Brannick and Ludmil T Zikatanov

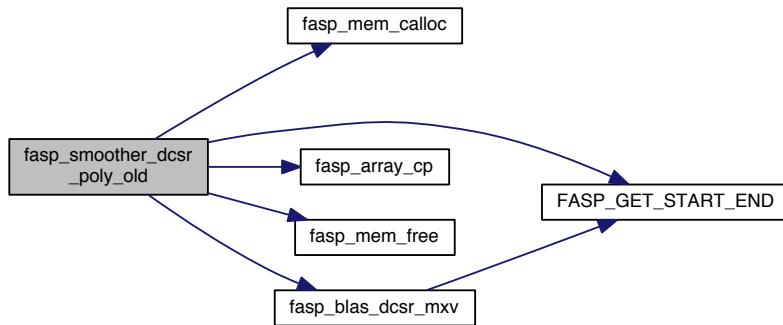
Date

06/28/2010

Modified by Chunsheng Feng, Zheng Li on 10/18/2012

Definition at line 148 of file smoother_csr_poly.c.

Here is the call graph for this function:

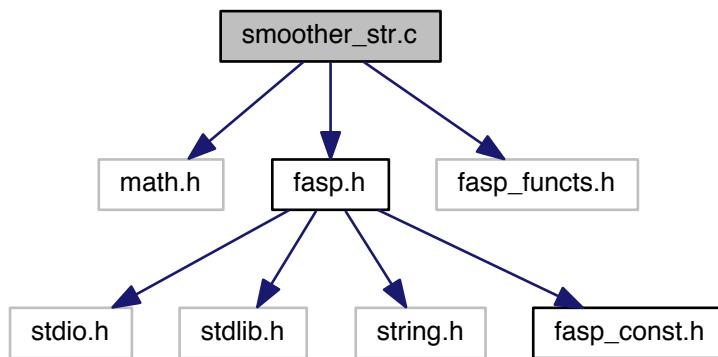


9.86 smoother_str.c File Reference

Smothers for dSTRmat matrices.

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

Include dependency graph for smoother_str.c:



Functions

- void `fasp_smoothen_dstr_jacobi (dSTRmat *A, dvector *b, dvector *u)`
Jacobi method as the smoother.
- void `fasp_smoothen_dstr_jacobi1 (dSTRmat *A, dvector *b, dvector *u, REAL *diaginv)`
Jacobi method as the smoother with diag_inv given.
- void `fasp_smoothen_dstr_gs (dSTRmat *A, dvector *b, dvector *u, INT order, INT *mark)`
Gauss-Seidel method as the smoother.
- void `fasp_smoothen_dstr_gs1 (dSTRmat *A, dvector *b, dvector *u, INT order, INT *mark, REAL *diaginv)`
Gauss-Seidel method as the smoother with diag_inv given.
- void `fasp_smoothen_dstr_gs_ascend (dSTRmat *A, dvector *b, dvector *u, REAL *diaginv)`
Gauss-Seidel method as the smoother in the ascending manner.
- void `fasp_smoothen_dstr_gs_descend (dSTRmat *A, dvector *b, dvector *u, REAL *diaginv)`
Gauss-Seidel method as the smoother in the descending manner.
- void `fasp_smoothen_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_smoothen_dstr_gs_cf (dSTRmat *A, dvector *b, dvector *u, REAL *diaginv, INT *mark, INT order)`
Gauss method as the smoother in the C-F manner.
- void `fasp_smoothen_dstr_sor (dSTRmat *A, dvector *b, dvector *u, INT order, INT *mark, REAL weight)`
SOR method as the smoother.
- void `fasp_smoothen_dstr_sor1 (dSTRmat *A, dvector *b, dvector *u, INT order, INT *mark, REAL *diaginv, REAL weight)`
SOR method as the smoother.
- void `fasp_smoothen_dstr_sor_ascend (dSTRmat *A, dvector *b, dvector *u, REAL *diaginv, REAL weight)`
SOR method as the smoother in the ascending manner.
- void `fasp_smoothen_dstr_sor_descend (dSTRmat *A, dvector *b, dvector *u, REAL *diaginv, REAL weight)`
SOR method as the smoother in the descending manner.
- void `fasp_smoothen_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_smoothen_dstr_sor_cf (dSTRmat *A, dvector *b, dvector *u, REAL *diaginv, INT *mark, INT order, REAL weight)`
SOR method as the smoother in the C-F manner.
- void `fasp_generate_diaginv_block (dSTRmat *A, ivec *neigh, dvector *diaginv, ivec *pivot)`
Generate inverse of diagonal block for block smoothers.
- void `fasp_smoothen_dstr_schwarz (dSTRmat *A, dvector *b, dvector *u, dvector *diaginv, ivec *pivot, ivec *neigh, ivec *order)`
Schwarz method as the smoother.

9.86.1 Detailed Description

Smoothers for `dSTRmat` matrices.

9.86.2 Function Documentation

9.86.2.1 void `fasp_generate_diaginv_block (dSTRmat * A, ivec * neigh, dvector * diaginv, ivec * pivot)`

Generate inverse of diagonal block for block smoothers.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>neigh</i>	Pointer to ivector: neighborhoods
<i>diaginv</i>	Pointer to dvector: the inverse of the diagonals
<i>pivot</i>	Pointer to ivector: the pivot of diagonal blocks

Author

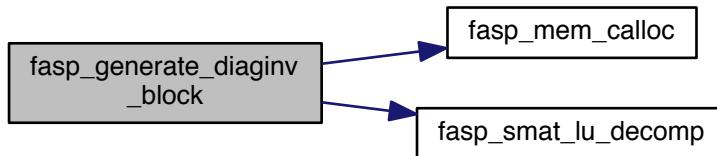
Xiaozhe Hu

Date

10/01/2011

Definition at line 1517 of file smoother_str.c.

Here is the call graph for this function:



9.86.2.2 void fasp_smoothen_dstr_gs (**dSTRmat** * *A*, **dvector** * *b*, **dvector** * *u*, INT *order*, INT * *mark*)

Gauss-Seidel method as the smoother.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns
<i>order</i>	Flag to indicate the order for smoothing If mark = NULL ASCEND 12: in ascending manner D ← ESCEND 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

<i>mark</i>	Pointer to the user-defined ordering(when order=0) or CF_marker array(when order!=0)
-------------	--

Author

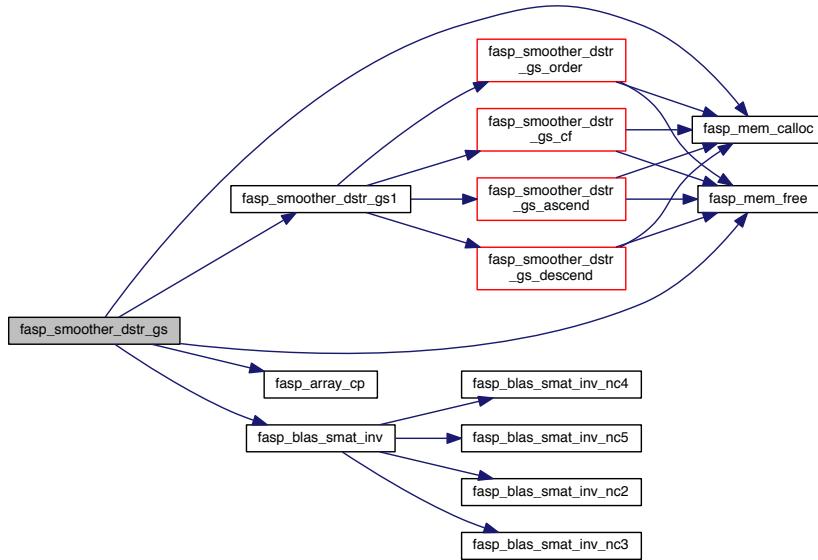
Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 202 of file smoother_str.c.

Here is the call graph for this function:



9.86.2.3 void fasp_smoothen_dstr_gs1 (dSTRmat * A, dvector * b, dvector * u, INT order, INT * mark, REAL * diaginv)

Gauss-Seidel method as the smoother with diag_inv given.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns
<i>order</i>	Flag to indicate the order for smoothing If mark = NULL ASCEND 12: in ascending manner D \leftarrow ESCEND 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
<i>mark</i>	Pointer to the user-defined ordering(when order=0) or CF_marker array(when order!=0)
<i>diaginv</i>	All the inverse matrices for all the diagonal block of A when (A->nc)>1, and NULL when (A->nc)=1

Author

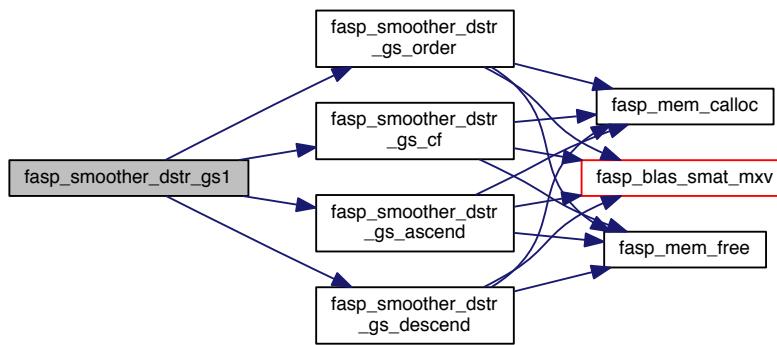
Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 261 of file smoother_str.c.

Here is the call graph for this function:



9.86.2.4 void fasp_smoothen_dstr_gs_ascend(dSTRmat * A, dvector * b, dvector * u, REAL * diaginv)

Gauss-Seidel method as the smoother in the ascending manner.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector : the right hand side
<i>u</i>	Pointer to dvector : the unknowns
<i>diaginv</i>	All the inverse matrices for all the diagonal block of <i>A</i> when (<i>A->nc</i>)>1, and NULL when (<i>A->nc</i>)=1

Author

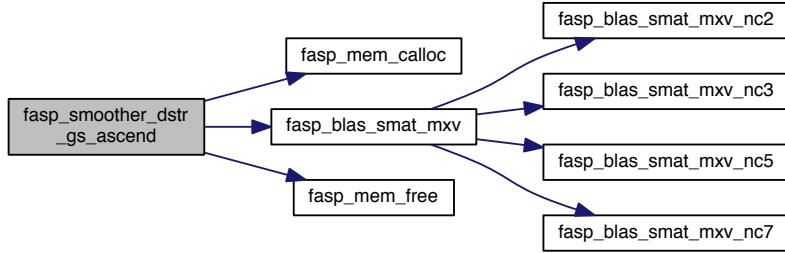
Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 306 of file smoother_str.c.

Here is the call graph for this function:



9.86.2.5 void fasp_smoothen_dstr_gs_cf(dSTRmat * A, dvector * b, dvector * u, REAL * diaginv, INT * mark, INT order)

Gauss method as the smoother in the C-F manner.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector : the right hand side
<i>u</i>	Pointer to dvector : the unknowns
<i>diaginv</i>	All the inverse matrices for all the diagonal block of <i>A</i> when (<i>A->nc</i>)>1, and NULL when (<i>A->nc</i>)=1
<i>mark</i>	Pointer to the user-defined order array
<i>order</i>	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

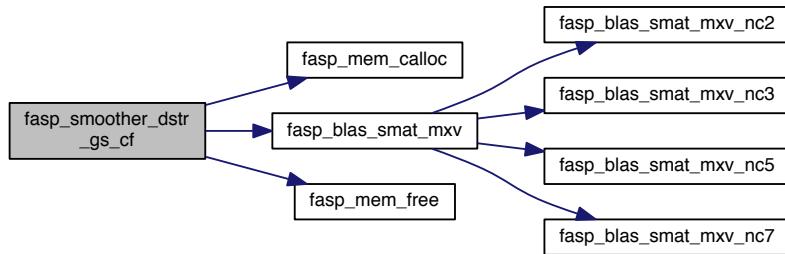
Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 660 of file smoother_str.c.

Here is the call graph for this function:



9.86.2.6 void fasp_smoothe_dstr_gs_descend (dSTRmat * A, dvector * b, dvector * u, REAL * diaginv)

Gauss-Seidel method as the smoother in the descending manner.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns
<i>diaginv</i>	All the inverse matrices for all the diagonal block of A when (A->nc)>1, and NULL when (A->nc)=1

Author

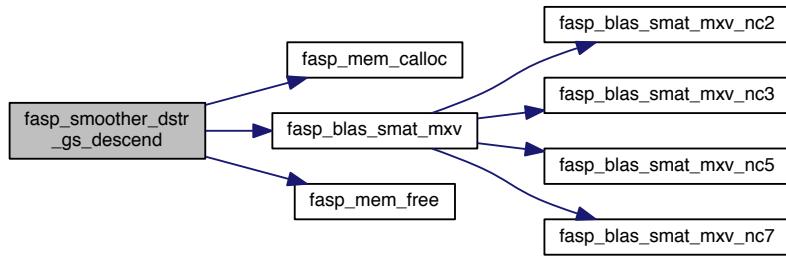
Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 421 of file smoother_str.c.

Here is the call graph for this function:



9.86.2.7 void fasp_smoothe_dstr_gs_order (dSTRmat * A, dvector * b, dvector * u, REAL * diaginv, INT * mark)

Gauss method as the smoother in the user-defined order.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns
<i>diaginv</i>	All the inverse matrices for all the diagonal block of A when (A->nc)>1, and NULL when (A->nc)=1
<i>mark</i>	Pointer to the user-defined order array

Author

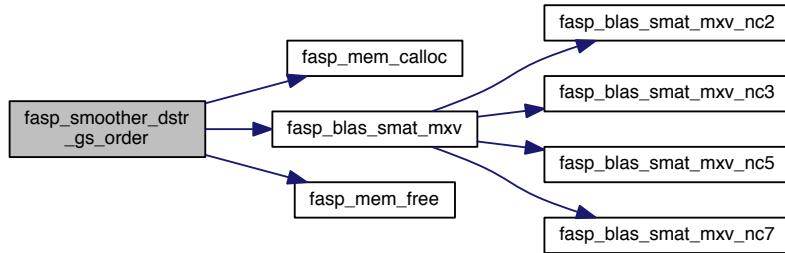
Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 538 of file smoother_str.c.

Here is the call graph for this function:



9.86.2.8 void fasp_smoothen_dstr_jacobi (dSTRmat * A, dvector * b, dvector * u)

Jacobi method as the smoother.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns

Author

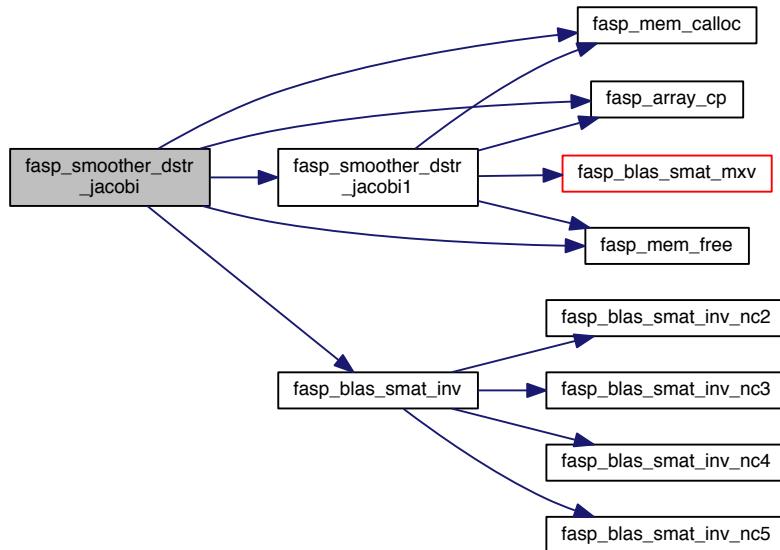
Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 31 of file smoother_str.c.

Here is the call graph for this function:



9.86.2.9 void fasp_smoothen_dstr_jacobi (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

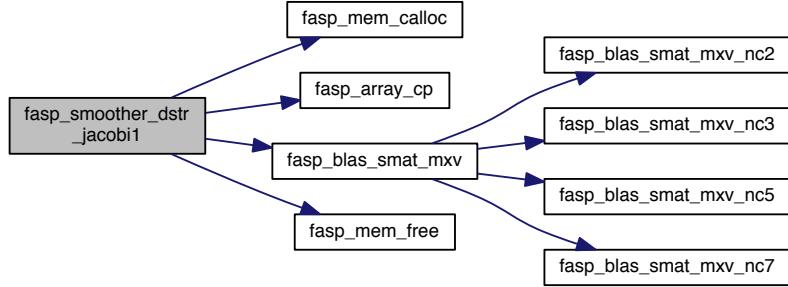
Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 79 of file smoother_str.c.

Here is the call graph for this function:



9.86.2.10 void fasp_smoothen_dstr_schwarz(dSTRmat * *A*, dvector * *b*, dvector * *u*, dvector * *diaginv*, ivecotor * *pivot*, ivecotor * *neigh*, ivecotor * *order*)

Schwarz method as the smoother.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns
<i>diaginv</i>	Pointer to dvector: the inverse of the diagonals
<i>pivot</i>	Pointer to ivecotor: the pivot of diagonal blocks
<i>neigh</i>	Pointer to ivecotor: neighborhoods
<i>order</i>	Pointer to ivecotor: the smoothing order

Author

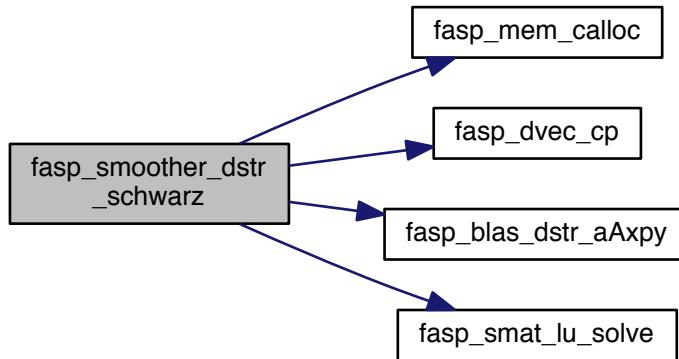
Xiaozhe Hu

Date

10/01/2011

Definition at line 1639 of file smoother_str.c.

Here is the call graph for this function:



9.86.2.11 void fasp_smoothen_dstr_sor (dSTRmat * A, dvector * b, dvector * u, INT order, INT * mark, REAL weight)

SOR method as the smoother.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns
<i>order</i>	Flag to indicate the order for smoothing If mark = NULL ASCEND 12: in ascending manner D ← ESCEND 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
<i>mark</i>	Pointer to the user-defined ordering(when order=0) or CF_marker array(when order!=0)
<i>weight</i>	Over-relaxation weight

Author

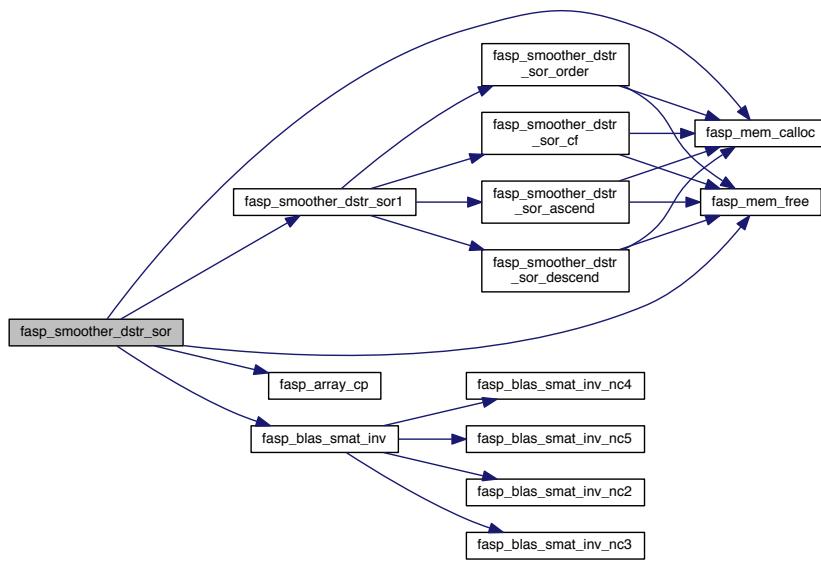
Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 851 of file smoother_str.c.

Here is the call graph for this function:



9.86.2.12 void fasp_smoothen_dstr_sor1 (dSTRmat * A, dvector * b, dvector * u, INT order, INT * mark, REAL * diaginv, 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 D-ESCAPE 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

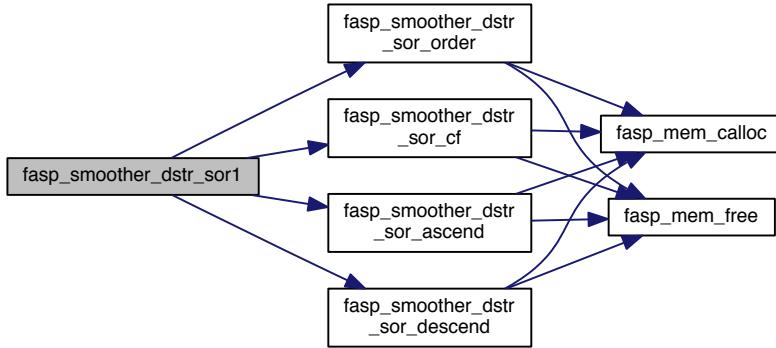
Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 912 of file `smoother_str.c`.

Here is the call graph for this function:



9.86.2.13 void fasp_smoothen_dstr_sor_ascend (dSTRmat * A, dvector * b, dvector * u, REAL * diaginv, REAL weight)

SOR method as the smoother in the ascending manner.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector : the right hand side
<i>u</i>	Pointer to dvector : the unknowns
<i>diaginv</i>	All the inverse matrices for all the diagonal block of <i>A</i> when (<i>A->nc</i>)>1, and NULL when (<i>A->nc</i>)=1
<i>weight</i>	Over-relaxation weight

Author

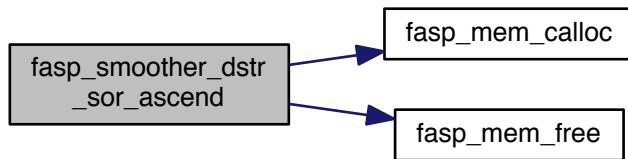
Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 958 of file smoother_str.c.

Here is the call graph for this function:



9.86.2.14 void fasp_smoothen_dstr_sor_cf(dSTRmat * A, dvector * b, dvector * u, REAL * diaginv, INT * mark, INT order, REAL weight)

SOR method as the smoother in the C-F manner.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns
<i>diaginv</i>	All the inverse matrices for all the diagonal block of A when (A->nc)>1, and NULL when (A->nc)=1
<i>mark</i>	Pointer to the user-defined order array
<i>order</i>	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
<i>weight</i>	Over-relaxation weight

Author

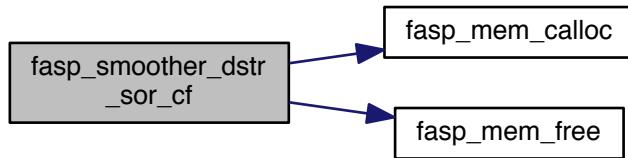
Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 1330 of file smoother_str.c.

Here is the call graph for this function:



9.86.2.15 void fasp_smoothen_dstr_sor_descend (dSTRmat * A, dvector * b, dvector * u, REAL * diaginv, REAL weight)

SOR method as the smoother in the descending manner.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector : the right hand side
<i>u</i>	Pointer to dvector : the unknowns
<i>diaginv</i>	All the inverse matrices for all the diagonal block of <i>A</i> when (<i>A->nc</i>)>1, and NULL when (<i>A->nc</i>)=1
<i>weight</i>	Over-relaxation weight

Author

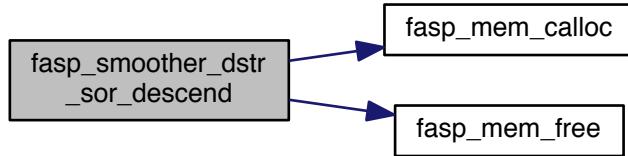
Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 1078 of file smoother_str.c.

Here is the call graph for this function:



9.86.2.16 void fasp_smoothen_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

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns
<i>diaginv</i>	All the inverse matrices for all the diagonal block of <i>A</i> when (<i>A->nc</i>)>1, and NULL when (<i>A->nc</i>)=1
<i>mark</i>	Pointer to the user-defined order array
<i>weight</i>	Over-relaxation weight

Author

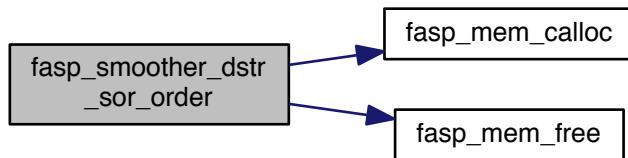
Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 1199 of file smoother_str.c.

Here is the call graph for this function:

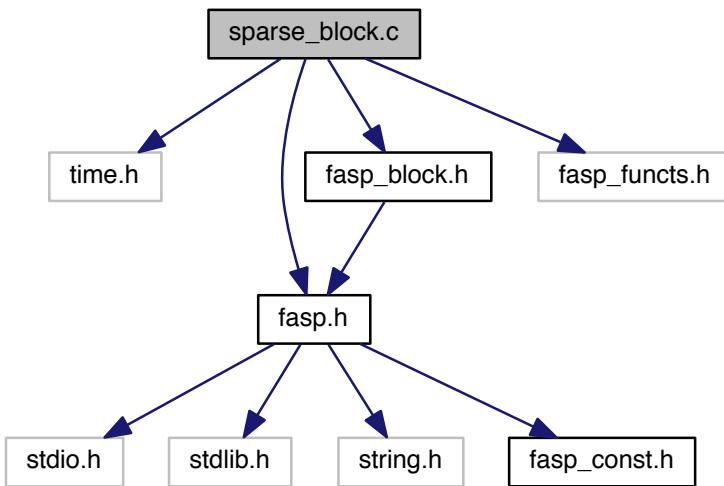


9.87 sparse_block.c File Reference

Sparse matrix block operations.

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

Include dependency graph for sparse_block.c:



Functions

- void [fasp_bdcsr_free \(block_dCSRmat *A\)](#)
Free block CSR sparse matrix data memory space.
- [SHORT fasp_dcsr_getblk \(dCSRmat *A, INT *Is, INT *Js, INT m, INT n, dCSRmat *B\)](#)
Get a sub CSR matrix of A with specified rows and columns.
- [SHORT fasp_dbsr_getblk \(dBSRmat *A, INT *Is, INT *Js, INT m, INT n, dBSRmat *B\)](#)
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

9.87.1 Detailed Description

Sparse matrix block operations.

9.87.2 Function Documentation

9.87.2.1 void [fasp_bdcsr_free \(block_dCSRmat * A \)](#)

Free block CSR sparse matrix data memory space.

Parameters

A	Pointer to the block_dCSRmat matrix
---	---

Author

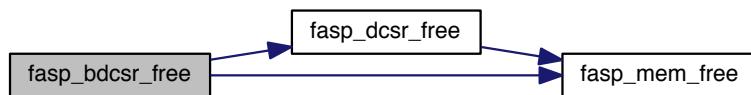
Xiaozhe Hu

Date

04/18/2014

Definition at line 30 of file `sparse_block.c`.

Here is the call graph for this function:



9.87.2.2 [SHORT fasp_dbsr_getblk \(dBSRmat * A, INT * Is, INT * Js, INT m, INT n, dBSRmat * B \)](#)

Get a sub BSR matrix of A with specified rows and columns.

Parameters

<i>A</i>	Pointer to <code>dBSRmat</code> BSR matrix
<i>B</i>	Pointer to <code>dBSRmat</code> BSR matrix
<i>Is</i>	Pointer to selected rows
<i>Js</i>	Pointer to selected columns
<i>m</i>	Number of selected rows
<i>n</i>	Number of selected columns

Returns

`FASP_SUCCESS` if succeeded, otherwise return error information.

Author

Shiquan Zhang, Xiaozhe Hu

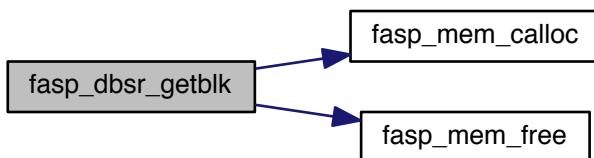
Date

12/25/2010

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

Definition at line 158 of file `sparse_block.c`.

Here is the call graph for this function:

**9.87.2.3 dCSRmat fasp_dbsr_getblk_dcsr (dBsrmat *A)**

get `dCSRmat` block from a `dBSRmat` matrix

Parameters

<code>*A</code>	Pointer to the BSR format matrix
-----------------	----------------------------------

Returns

`dCSRmat` matrix if succeed, NULL if fail

Author

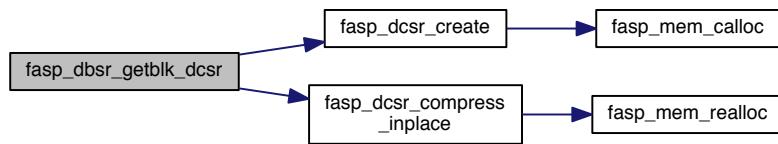
Xiaozhe Hu

Date

03/16/2012

Definition at line 254 of file sparse_block.c.

Here is the call graph for this function:



9.87.2.4 `dCSRmat fasp_dbsr_Linfinity_dcsr (dBsrmat *A)`

get `dCSRmat` from a `dBsrmat` matrix using L_infinity norm of each small block

Parameters

<code>*A</code>	Pointer to the BSR format matrix
-----------------	----------------------------------

Returns

`dCSRmat` matrix if succeed, NULL if fail

Author

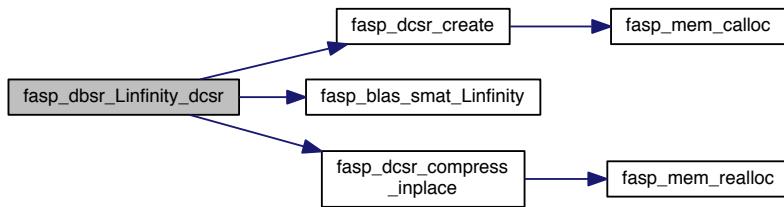
Xiaozhe Hu

Date

05/25/2014

Definition at line 310 of file sparse_block.c.

Here is the call graph for this function:

**9.87.2.5 SHORT fasp_dcsr_getblk(dCSRmat * A, INT * ls, INT * Js, INT m, INT n, dCSRmat * B)**

Get a sub CSR matrix of A with specified rows and columns.

Parameters

<i>A</i>	Pointer to dCSRmat matrix
<i>B</i>	Pointer to dCSRmat matrix
<i>ls</i>	Pointer to selected rows
<i>Js</i>	Pointer to selected columns
<i>m</i>	Number of selected rows
<i>n</i>	Number of selected columns

Returns

FASP_SUCCESS if succeeded, otherwise return error information.

Author

Shiquan Zhang, Xiaozhe Hu

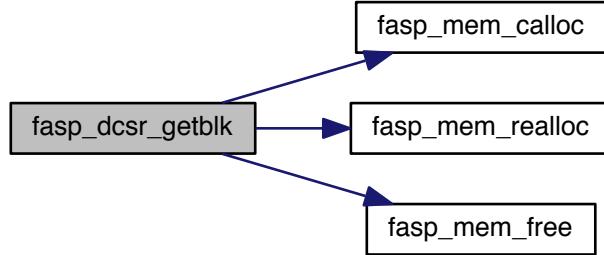
Date

12/25/2010

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

Definition at line 65 of file sparse_block.c.

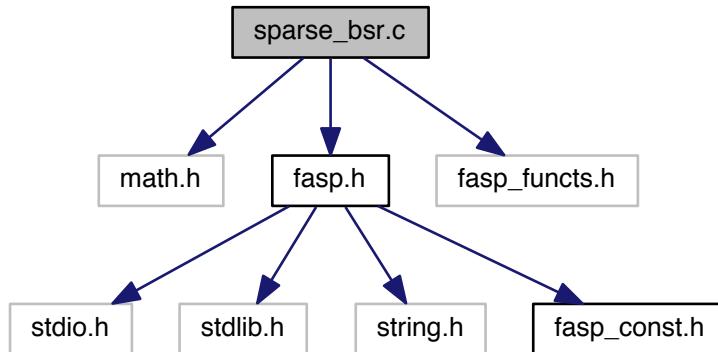
Here is the call graph for this function:



9.88 sparse_bsr.c File Reference

Sparse matrix operations for `dBSRmat` matrices.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
Include dependency graph for sparse_bsr.c:
```



Functions

- `dBSRmat fasp_dbsr_create (INT ROW, INT COL, INT NNZ, INT nb, INT storage_manner)`
Create BSR sparse matrix data memory space.
- `void fasp_dbsr_alloc (INT ROW, INT COL, INT NNZ, INT nb, INT storage_manner, dBSPmat *A)`

- **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 $B := DL \cdot A \cdot DU$. We decompose each diagonal block of A into LDU form and $DL = \text{diag}(L^{-1})$ and $DU = \text{diag}(U^{-1})$.
- **dBSRmat fasp_dbsr_diagLU2 (dBSRmat *A, REAL *DL, REAL *DU)**
Compute $B := DL \cdot A \cdot DU$. We decompose each diagonal block of A into LDU form and $DL = \text{diag}(L^{-1})$ and $DU = \text{diag}(U^{-1})$.

9.88.1 Detailed Description

Sparse matrix operations for [dBSRmat](#) matrices.

9.88.2 Function Documentation

9.88.2.1 void fasp_dbsr_alloc (INT ROW, INT COL, INT NNZ, INT nb, INT storage_manner, dBSRmat * A)

Allocate memory space for BSR format sparse matrix.

Parameters

<i>ROW</i>	Number of rows of block
<i>COL</i>	Number of columns of block

<i>NNZ</i>	Number of nonzero blocks
<i>nb</i>	Dimension of each block
<i>storage_manner</i>	Storage manner for each sub-block
<i>A</i>	Pointer to new dBSRmat matrix

Author

Xiaozhe Hu

Date

10/26/2010

Definition at line 86 of file `sparse_bsr.c`.

Here is the call graph for this function:



9.88.2.2 void `fasp_dbsr_cp` (`dBSRmat * A, dBSRmat * B`)

copy a [dCSRmat](#) to a new one $B=A$

Parameters

<i>A</i>	Pointer to the dBSRmat matrix
<i>B</i>	Pointer to the dBSRmat matrix

Author

Xiaozhe Hu

Date

08/07/2011

Definition at line 180 of file `sparse_bsr.c`.

9.88.2.3 `dBSRmat fasp_dbsr_create` (`INT ROW, INT COL, INT NNZ, INT nb, INT storage_manner`)

Create BSR sparse matrix data memory space.

Parameters

<i>ROW</i>	Number of rows of block
<i>COL</i>	Number of columns of block
<i>NNZ</i>	Number of nonzero blocks
<i>nb</i>	Dimension of each block
<i>storage_manner</i>	Storage manner for each sub-block

Returns

A The new [dBSRmat](#) matrix

Author

Xiaozhe Hu

Date

10/26/2010

Definition at line 35 of file `sparse_bsr.c`.

Here is the call graph for this function:

**9.88.2.4 [dBSRmat](#) `fasp_dbsr_diaginv(dBSRmat * A)`**

Compute $B := D^{-1} \cdot A$, where 'D' is the block diagonal part of A.

Parameters

<i>A</i>	Pointer to the dBSRmat matrix
----------	---

Author

Zhiyang Zhou

Date

2010/10/26

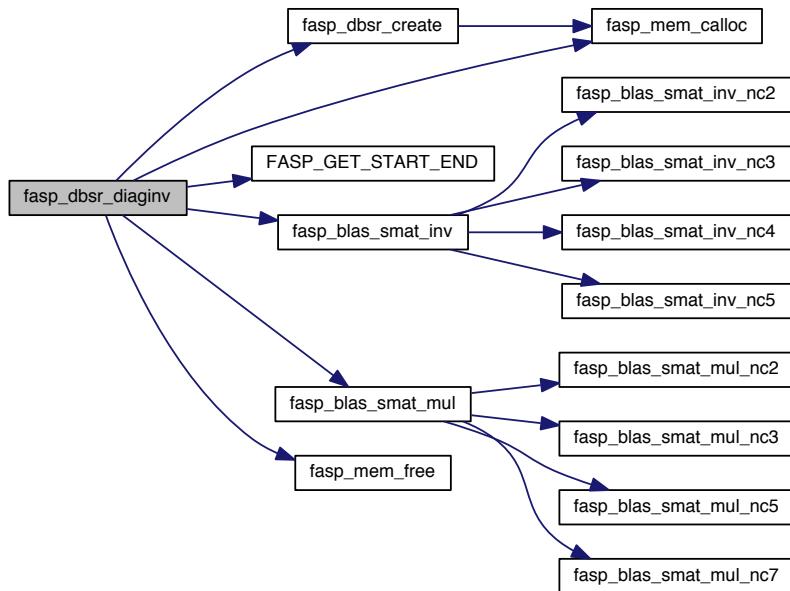
Note

Works for general nb (Xiaozhe)

Modified by Chunsheng Feng, Zheng Li on 08/25/2012

Definition at line 496 of file sparse_bsr.c.

Here is the call graph for this function:



9.88.2.5 `dBSRmat fasp_dbsr_diaginv2(dBSRmat * A, REAL * diaginv)`

Compute $B := D^{-1} \cdot A$, where 'D' is the block diagonal part of A.

Parameters

<code>A</code>	Pointer to the <code>dBSRmat</code> matrix
<code>diaginv</code>	Pointer to the inverses of all the diagonal blocks

Author

Zhiyang Zhou

Date

2010/11/07

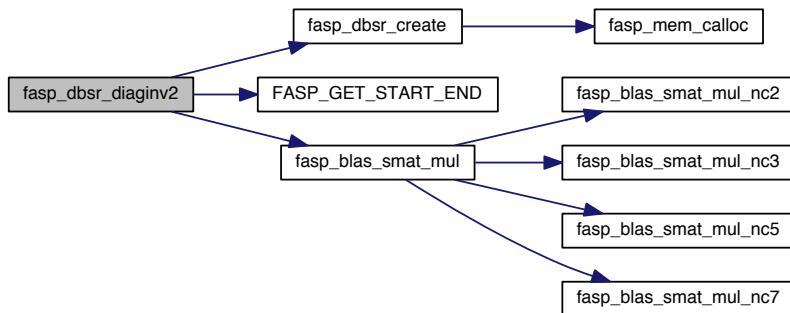
Note

Works for general nb (Xiaozhe)

Modified by Chunsheng Feng, Zheng Li on 08/25/2012

Definition at line 660 of file sparse_bsr.c.

Here is the call graph for this function:



9.88.2.6 dBRSRmat fasp_dbsr_diaginv3 (dBRSRmat * A, REAL * diaginv)

Compute $B := D^{-1} \cdot A$, where 'D' is the block diagonal part of A.

Parameters

<i>A</i>	Pointer to the dBRSRmat matrix
<i>diaginv</i>	Pointer to the inverses of all the diagonal blocks

Returns

BSR matrix after diagonal scaling

Author

Xiaozhe Hu

Date

12/25/2010

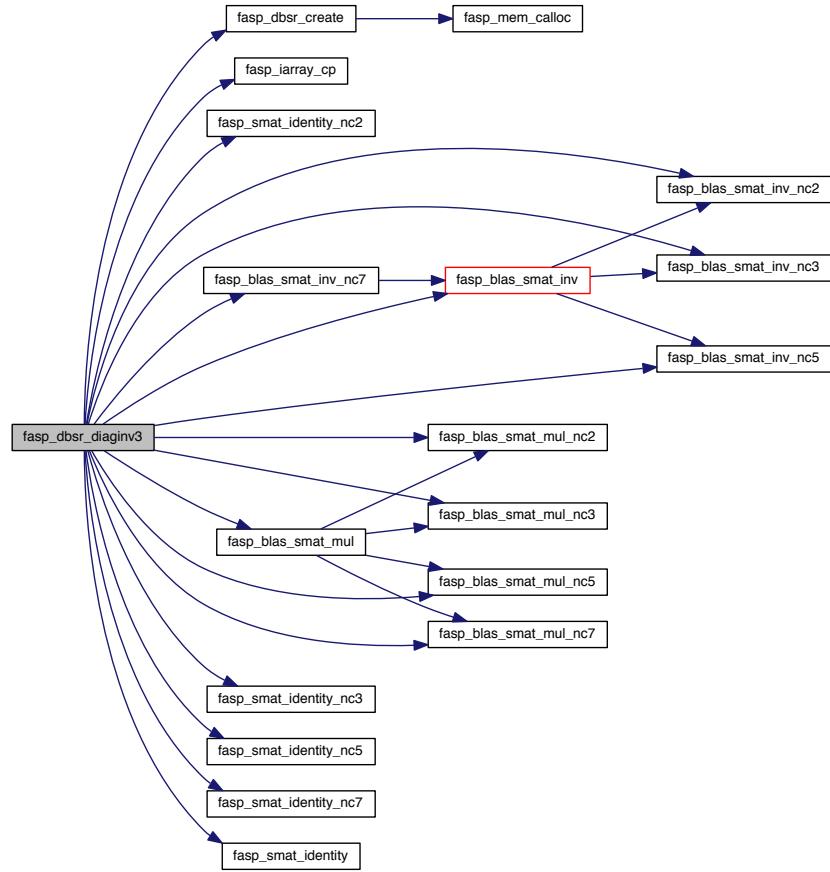
Note

Works for general nb (Xiaozhe)

Modified by Xiaozhe Hu on 05/26/2012

Definition at line 763 of file sparse_bsr.c.

Here is the call graph for this function:



9.88.2.7 dBSRmat fasp_dbsr_diaginv4 (dBSRmat * A, REAL * diaginv)

Compute $B := D^{-1} \cdot A$, where 'D' is the block diagonal part of A.

Parameters

<i>A</i>	Pointer to the dBSRmat matrix
<i>diaginv</i>	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

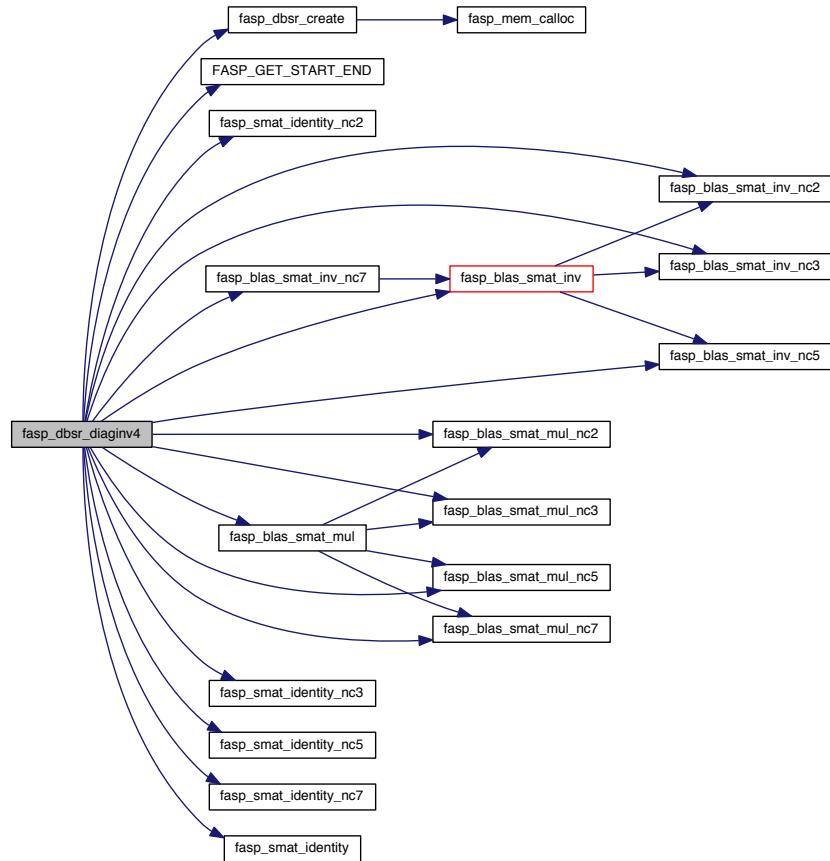
Date

03/12/2011

Modified by Chunsheng Feng, Zheng Li on 08/26/2012

Definition at line 1121 of file sparse_bsr.c.

Here is the call graph for this function:



9.88.2.8 dBSRmat fasp_dbsr_diagLU (dBSRmat *A, REAL *DL, REAL *DU)

Compute $B := DL \cdot A \cdot DU$. We decompose each diagonal block of A into LDU form and $DL = \text{diag}(L^{\wedge\{-1\}})$ and $DU = \text{diag}(U^{\wedge\{-1\}})$.

Parameters

<i>A</i>	Pointer to the <code>dBSRmat</code> matrix
<i>DL</i>	Pointer to the $\text{diag}(L^{-1})$
<i>DU</i>	Pointer to the $\text{diag}(U^{-1})$

Returns

BSR matrix after scaling

Author

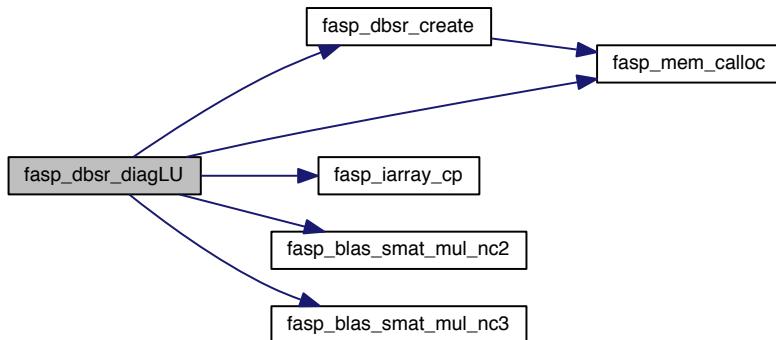
Xiaozhe Hu

Date

04/02/2014

Definition at line 1448 of file `sparse_bsr.c`.

Here is the call graph for this function:

**9.88.2.9 `dBSRmat fasp_dbsr_diagLU2 (dBSRmat * A, REAL * DL, REAL * DU)`**

Compute $B := DL \cdot A \cdot DU$. We decompose each diagonal block of A into LDU form and $DL = \text{diag}(L^{-1})$ and $DU = \text{diag}(U^{-1})$.

Parameters

<i>A</i>	Pointer to the <code>dBSRmat</code> matrix
<i>DL</i>	Pointer to the $\text{diag}(L^{-1})$

<i>DU</i>	Pointer to the $\text{diag}(U^{-1})$
-----------	--------------------------------------

Returns

BSR matrix after scaling

Author

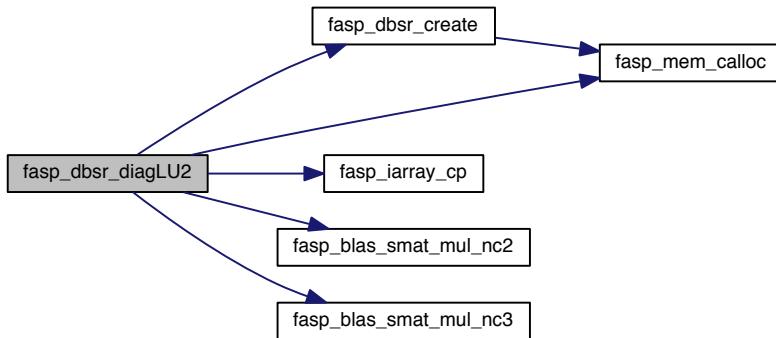
Zheng Li, Xiaozhe Hu

Date

06/17/2014

Definition at line 1676 of file sparse_bsr.c.

Here is the call graph for this function:

**9.88.2.10 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

<i>A</i>	Pointer to the BSR matrix
----------	---------------------------

Author

Xiaozhe Hu

Date

03/10/2011

Author

Chunsheng Feng, Zheng Li

Date

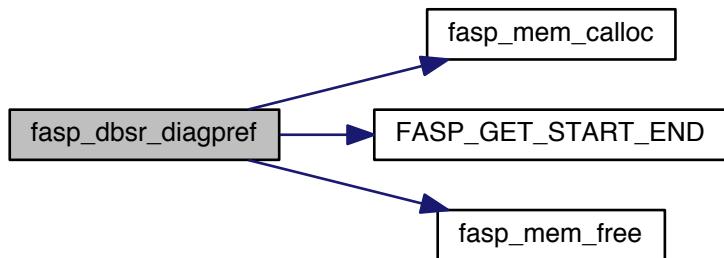
09/02/2012

Note

Reordering is done in place.

Definition at line 291 of file sparse_bsr.c.

Here is the call graph for this function:

**9.88.2.11 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 132 of file sparse_bsr.c.

Here is the call graph for this function:



9.88.2.12 fasp_dbsr_getdiag (INT *n*, dBRSRmat * *A*, REAL * *diag*)

Abstract the diagonal blocks of a BSR matrix.

Parameters

<i>n</i>	Number of blocks to get
<i>A</i>	Pointer to the 'dBRSRmat' type matrix
<i>diag</i>	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 1412 of file sparse_bsr.c.

9.88.2.13 dvector fasp_dbsr_getdiaginv (dBRSRmat * *A*)

Get D^{-1} of matrix *A*.

Parameters

<i>A</i>	Pointer to the dBRSRmat matrix
----------	--------------------------------

Author

Xiaozhe Hu

Date

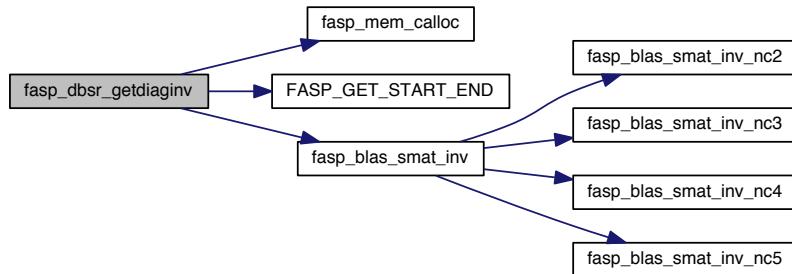
02/19/2013

Note

Works for general nb (Xiaozhe)

Definition at line 392 of file sparse_bsr.c.

Here is the call graph for this function:



9.88.2.14 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 157 of file sparse_bsr.c.

9.88.2.15 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

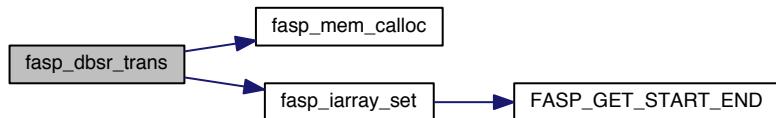
Date

2011/06/08

Modified by Xiaozhe Hu (08/06/2011)

Definition at line 207 of file sparse_bsr.c.

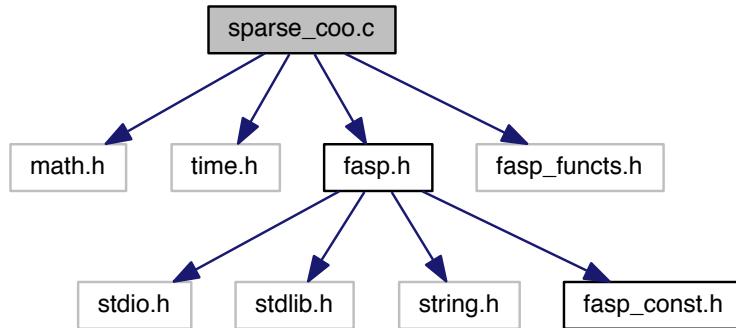
Here is the call graph for this function:



9.89 sparse_coo.c File Reference

Sparse matrix operations for [dCOOmat](#) matrices.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
Include dependency graph for sparse_coo.c:
```



Functions

- [dCOOmat fasp_dcoo_create \(INT m, INT n, INT nnz\)](#)

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 , INT $offset$)
Re-index a REAL matrix in IJ format to make the index starting from 0 or 1.

9.89.1 Detailed Description

Sparse matrix operations for `dCOOmat` matrices.

9.89.2 Function Documentation

9.89.2.1 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 <code>dCSRmat</code> matrix

Author

Xiaozhe Hu

Date

03/25/2013

Definition at line 62 of file `sparse_coo.c`.

Here is the call graph for this function:



9.89.2.2 dCOOmat `fasp_dcoo_create` (INT m , INT n , INT nnz)

Create IJ sparse matrix data memory space.

Parameters

<i>m</i>	Number of rows
<i>n</i>	Number of columns
<i>nnz</i>	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`.

Here is the call graph for this function:



9.89.2.3 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`.

Here is the call graph for this function:



9.89.2.4 void fasp_dcoo_shift (dCOOmat * A, INT offset)

Re-index a REAL matrix in IJ format to make the index starting from 0 or 1.

Parameters

<i>A</i>	Pointer to IJ matrix
<i>offset</i>	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.

Here is the call graph for this function:

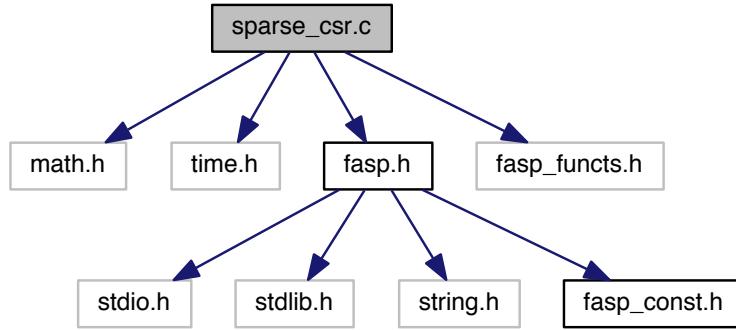


9.90 sparse_csr.c File Reference

Sparse matrix operations for [dCSRmat](#) matrices.

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

Include dependency graph for sparse_csr.c:



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. $A_{perm} = PAP'$ 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 $\text{abs}(aij) \leq dtol$.

- SHORT [fasp_dcsr_compress_inplace](#) ([dCSRmat](#) *A, REAL dtol)

Compress a CSR matrix A IN PLACE by dropping small entries $\text{abs}(aij) \leq dtol$.

- 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.

9.90.1 Detailed Description

Sparse matrix operations for [dCSRmat](#) matrices.

9.90.2 Function Documentation

9.90.2.1 void [fasp_dcsr_alloc](#) (const INT m, const INT n, const INT nnz, [dCSRmat](#) * A)

Allocate CSR sparse matrix memory space.

Parameters

<i>m</i>	Number of rows
<i>n</i>	Number of columns
<i>nnz</i>	Number of nonzeros
<i>A</i>	Pointer to the dCSRmat matrix

Author

Chensong Zhang

Date

2010/04/06

Definition at line 125 of file sparse_csr.c.

Here is the call graph for this function:



9.90.2.2 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 $\text{abs}(a_{ij}) \leq \text{dtol}$.

Parameters

A	Pointer to dCSRmat CSR matrix
B	Pointer to dCSRmat CSR matrix
dtol	Drop tolerance

Author

Shiquan Zhang

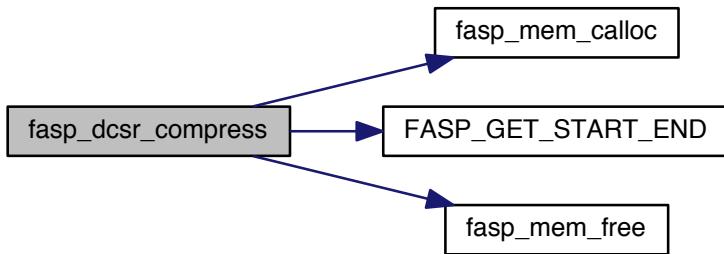
Date

03/10/2010

Modified by Chunsheng Feng, Zheng Li on 08/25/2012

Definition at line 955 of file sparse_csr.c.

Here is the call graph for this function:



9.90.2.3 **SHORT fasp_dcsr_compress_inplace (dCSRmat * A, REAL dtol)**

Compress a CSR matrix A IN PLACE by dropping small entries $\text{abs}(a_{ij}) \leq dtol$.

Parameters

<i>A</i>	Pointer to dCSRmat CSR matrix
<i>dtol</i>	Drop tolerance

Author

Xiaozhe Hu

Date

12/25/2010

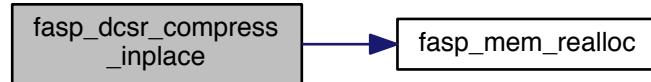
Modified by Chensong Zhang on 02/21/2013

Note

This routine can be modified for filtering.

Definition at line 1035 of file sparse_csr.c.

Here is the call graph for this function:



9.90.2.4 void fasp_dcsr_cp (dCSRmat * A, dCSRmat * B)

copy a [dCSRmat](#) to a new one B=A

Parameters

<i>A</i>	Pointer to the dCSRmat matrix
<i>B</i>	Pointer to the dCSRmat matrix

Author

Chensong Zhang

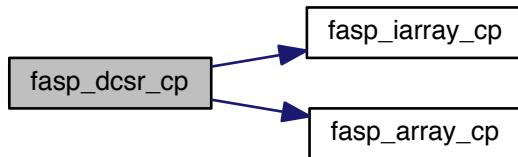
Date

04/06/2010

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

Definition at line 721 of file sparse_csr.c.

Here is the call graph for this function:



9.90.2.5 dCSRmat fasp_dcsr_create (const INT *m*, const INT *n*, const INT *nnz*)

Create CSR sparse matrix data memory space.

Parameters

<i>m</i>	Number of rows
<i>n</i>	Number of columns
<i>nnz</i>	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.

Here is the call graph for this function:



9.90.2.6 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

<i>A</i>	Pointer to the matrix to be re-ordered
----------	--

Author

Zhiyang Zhou

Date

09/09/2010

Author

Chunsheng Feng, Zheng Li

Date

09/02/2012

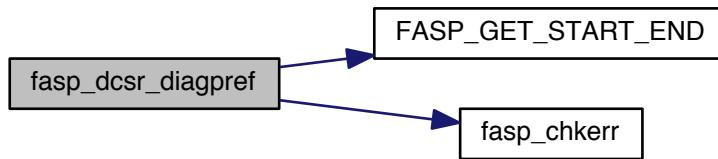
Note

Reordering is done in place.

Modified by Chensong Zhang on Dec/21/2012

Definition at line 551 of file sparse_csr.c.

Here is the call graph for this function:



9.90.2.7 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.

Here is the call graph for this function:



9.90.2.8 void fasp_dcsr_getcol (const INT *n*, dCSRmat * *A*, REAL * *col*)

Get the *n*-th column of a CSR matrix *A*.

Parameters

<i>n</i>	Index of a column of <i>A</i> ($0 \leq n \leq A.col-1$)
<i>A</i>	Pointer to dCSRmat CSR matrix
<i>col</i>	Pointer to the column

Author

Xiaozhe Hu

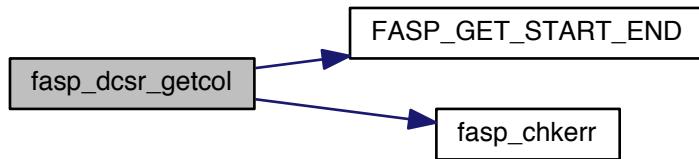
Date

11/07/2009

Modified by Chunsheng Feng, Zheng Li on 07/08/2012

Definition at line 472 of file `sparse_csr.c`.

Here is the call graph for this function:



9.90.2.9 void fasp_dcsr_getdiag (INT *n*, dCSRmat * *A*, dvector * *diag*)

Get first *n* diagonal entries of a CSR matrix *A*.

Parameters

<i>n</i>	Number of diagonal entries to get (if <i>n</i> =0, then get all diagonal entries)
<i>A</i>	Pointer to dCSRmat CSR matrix
<i>diag</i>	Pointer to the diagonal as a dvector

Author

Chensong Zhang

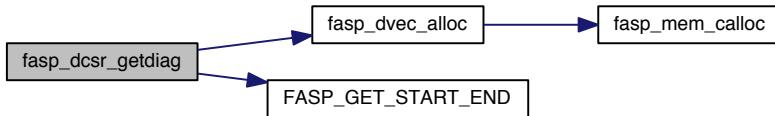
Date

05/20/2009

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

Definition at line 408 of file sparse_csr.c.

Here is the call graph for this function:



9.90.2.10 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

<i>A</i>	Input dCSRmat
<i>flags</i>	flags for the independent group
<i>groups</i>	Return group numbers

Author

Chunsheng Feng

Date

09/15/2012

Definition at line 1263 of file sparse_csr.c.

9.90.2.11 void fasp_dcsr_null (dCSRmat * A)

Initialize CSR sparse matrix.

Parameters

<i>A</i>	Pointer to the dCSRmat matrix
----------	---

Author

Chensong Zhang

Date

2010/04/03

Definition at line 204 of file sparse_csr.c.

9.90.2.12 dCSRmat fasp_dcsr_perm (dCSRmat * A, INT * P)

Apply permutation of A, i.e. $A_{perm} = PAP'$ by the orders given in P.

Parameters

<i>A</i>	Pointer to the original dCSRmat matrix
<i>P</i>	Pointer to orders

Returns

The new ordered [dCSRmat](#) matrix if succeed, NULL if fail

Author

Shiquan Zhang

Date

03/10/2010

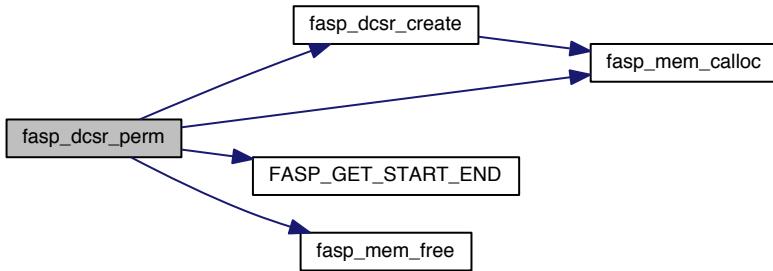
Note

$P[i] = k$ means k-th row and column become i-th row and column!

Modified by Chunsheng Feng, Zheng Li on 07/12/2012

Definition at line 245 of file sparse_csr.c.

Here is the call graph for this function:



9.90.2.13 SHORT fasp_dcsr_regdiag (dCSRmat * A, REAL value)

Regularize diagonal entries of a CSR sparse matrix.

Parameters

<i>A</i>	Pointer to the dCSRmat matrix
<i>value</i>	Set a value on diag(<i>A</i>) 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 657 of file [sparse_csr.c](#).

9.90.2.14 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

<i>A</i>	Pointer to CSR matrix
<i>offset</i>	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 1083 of file [sparse_csr.c](#).

Here is the call graph for this function:

**9.90.2.15 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

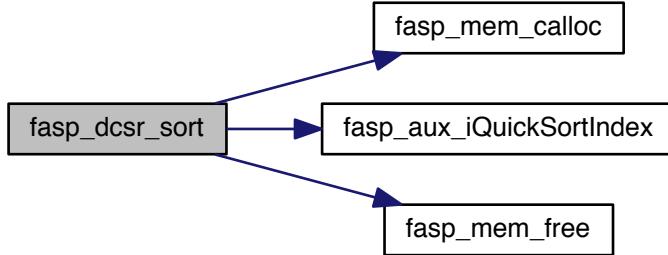
Shiquan Zhang

Date

06/10/2010

Definition at line 356 of file `sparse_csr.c`.

Here is the call graph for this function:



9.90.2.16 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

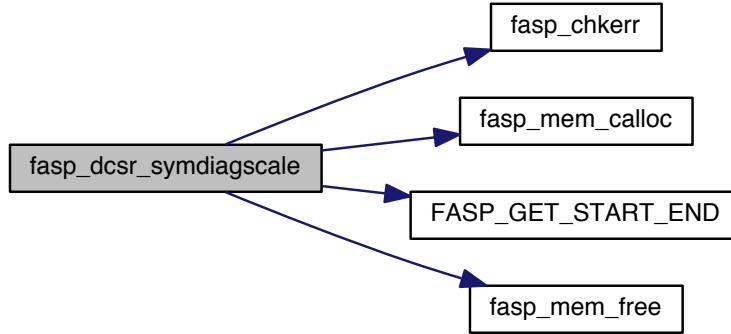
Date

01/31/2011

Modified by Chunsheng Feng, Zheng Li on 07/11/2012

Definition at line 1144 of file `sparse_csr.c`.

Here is the call graph for this function:



9.90.2.17 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

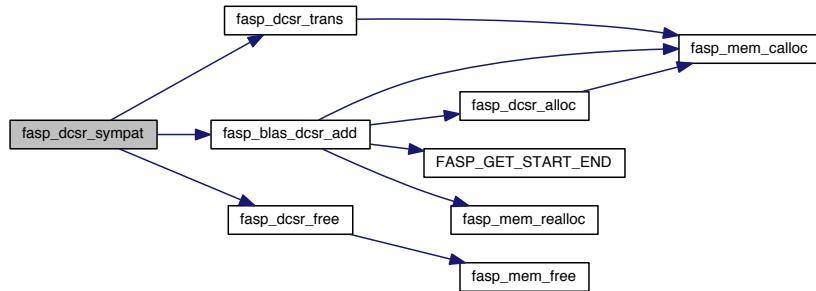
Xiaozhe Hu

Date

03/21/2011

Definition at line 1230 of file sparse_csr.c.

Here is the call graph for this function:

**9.90.2.18 void fasp_dCSRmat_trans (dCSRmat * A, dCSRmat * AT)**Find transpose of [dCSRmat](#) matrix A.**Parameters**

<i>A</i>	Pointer to the dCSRmat matrix
<i>AT</i>	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 824 of file sparse_csr.c.

Here is the call graph for this function:



9.90.2.19 void fasp_icsr_cp (iCSRmat * *A*, iCSRmat * *B*)

Copy a [iCSRmat](#) to a new one *B*=*A*.

Parameters

<i>A</i>	Pointer to the iCSRmat matrix
<i>B</i>	Pointer to the iCSRmat matrix

Author

Chensong Zhang

Date

05/16/2013

Definition at line 696 of file sparse_csr.c.

Here is the call graph for this function:



9.90.2.20 iCSRmat fasp_icsr_create (const INT *m*, const INT *n*, const INT *nnz*)

Create CSR sparse matrix data memory space.

Parameters

<i>m</i>	Number of rows
<i>n</i>	Number of columns
<i>nnz</i>	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.

Here is the call graph for this function:



9.90.2.21 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`.

Here is the call graph for this function:



9.90.2.22 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`.

9.90.2.23 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

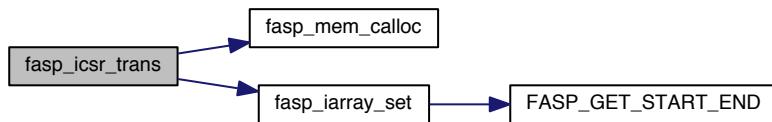
Date

04/06/2010

Modified by Chunsheng Feng, Zheng Li on 06/20/2012

Definition at line 748 of file `sparse_csr.c`.

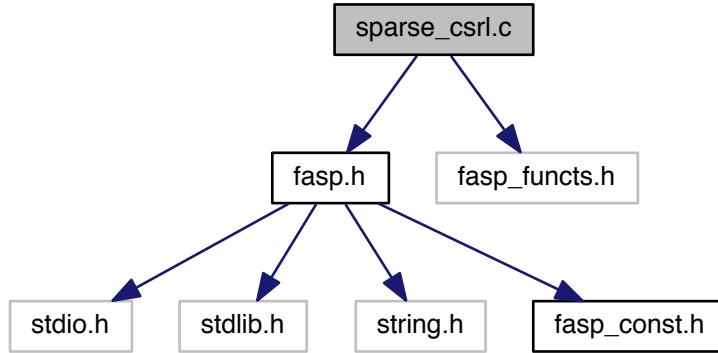
Here is the call graph for this function:



9.91 sparse_csr.c File Reference

Sparse matrix operations for [dCSRmat](#) matrices.

```
#include "fasp.h"
#include "fasp_functs.h"
Include dependency graph for sparse_csrl.c:
```



Functions

- `dCSRMat * fasp_dcsr_create (INT num_rows, INT num_cols, INT num_nonzeros)`
Create a `dCSRMat` object.
- `void fasp_dcsr_free (dCSRMat *A)`
Destroy a `dCSRMat` object.

9.91.1 Detailed Description

Sparse matrix operations for `dCSRMat` 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.

9.91.2 Function Documentation

9.91.2.1 `dCSRMat * fasp_dcsr_create (INT num_rows, INT num_cols, INT num_nonzeros)`

Create a `dCSRMat` object.

Parameters

<i>num_rows</i>	Number of rows
<i>num_cols</i>	Number of cols
<i>num_nonzeros</i>	Number of nonzero entries

Author

Zhiyang Zhou

Date

01/07/2001

Definition at line 30 of file sparse_csr1.c.

Here is the call graph for this function:



9.91.2.2 void fasp_dcsr1_free (dCSRMat * A)

Destroy a [dCSRMat](#) object.

Parameters

<i>A</i>	Pointer to the dCSRMat type matrix
----------	--

Author

Zhiyang Zhou

Date

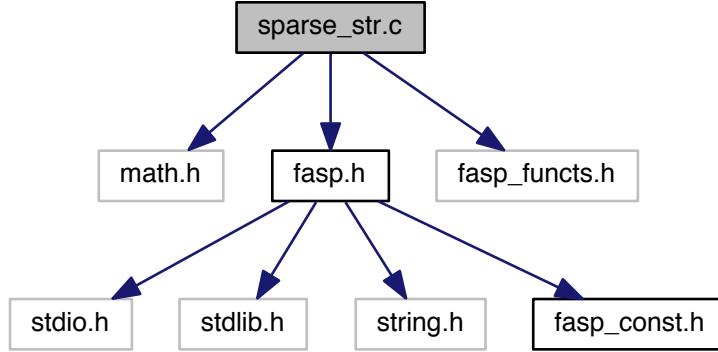
01/07/2011

Definition at line 58 of file sparse_csr1.c.

9.92 sparse_str.c File Reference

Sparse matrix operations for [dSTRmat](#) matrices.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
Include dependency graph for sparse_str.c:
```



Functions

- void [fasp_dstr_null](#) (`dSTRmat *A`)

Initialize sparse matrix on structured grid.
- [dSTRmat fasp_dstr_create](#) (`INT nx, INT ny, INT nz, INT nc, INT nband, INT *offsets`)

Create STR sparse matrix data memory space.
- void [fasp_dstr_alloc](#) (`INT nx, INT ny, INT nz, INT nxy, INT ngrid, INT nband, INT nc, INT *offsets, dSTRmat *A`)

Allocate STR sparse matrix memory space.
- void [fasp_dstr_free](#) (`dSTRmat *A`)

Free STR sparse matrix data memory space.
- void [fasp_dstr_cp](#) (`dSTRmat *A, dSTRmat *A1`)

Copy a `dSTRmat` to a new one $A1=A$.

9.92.1 Detailed Description

Sparse matrix operations for `dSTRmat` matrices.

9.92.2 Function Documentation

9.92.2.1 void [fasp_dstr_alloc](#) (`INT nx, INT ny, INT nz, INT nxy, INT ngrid, INT nband, INT nc, INT * offsets, dSTRmat * A`)

Allocate STR sparse matrix memory space.

Parameters

<i>nx</i>	Number of grids in x direction
<i>ny</i>	Number of grids in y direction
<i>nz</i>	Number of grids in z direction
<i>nxy</i>	Number of grids in x-y plane
<i>ngrid</i>	Number of grids
<i>nband</i>	Number of off-diagonal bands
<i>nc</i>	Number of components
<i>offsets</i>	Shift from diagonal
<i>A</i>	Pointer to the dSTRmat matrix

Author

Shiquan Zhang, Xiaozhe Hu

Date

05/17/2010

Definition at line 107 of file sparse_str.c.

Here is the call graph for this function:



9.92.2.2 void fasp_dstr_cp(dSTRmat * A, dSTRmat * A1)

Copy a [dSTRmat](#) to a new one A1=A.

Parameters

<i>A</i>	Pointer to the dSTRmat matrix
<i>A1</i>	Pointer to the dSTRmat matrix

Author

Zhiyang Zhou

Date

04/21/2010

Definition at line 179 of file sparse_str.c.

9.92.2.3 dSTRmat fasp_dstr_create (INT *nx*, INT *ny*, INT *nz*, INT *nc*, INT * *nband*, INT * *offsets*)

Create STR sparse matrix data memory space.

Parameters

<i>nx</i>	Number of grids in x direction
<i>ny</i>	Number of grids in y direction
<i>nz</i>	Number of grids in z direction
<i>nc</i>	Number of components
<i>nband</i>	Number of off-diagonal bands
<i>offsets</i>	Shift from diagonal

Returns

The [dSTRmat](#) matrix

Author

Shiquan Zhang, Xiaozhe Hu

Date

05/17/2010

Definition at line 56 of file sparse_str.c.

Here is the call graph for this function:



9.92.2.4 void fasp_dstr_free(dSTRmat * A)

Free STR sparse matrix data memory space.

Parameters

<i>A</i>	Pointer to the dSTRmat matrix
----------	---

Author

Shiquan Zhang, Xiaozhe Hu

Date

05/17/2010

Definition at line 150 of file `sparse_str.c`.

Here is the call graph for this function:



9.92.2.5 void `fasp_dstr_null(dSTRmat * A)`

Initialize sparse matrix on structured grid.

Parameters

<code>A</code>	Pointer to the <code>dSTRmat</code> matrix
----------------	--

Author

Shiquan Zhang, Xiaozhe Hu

Date

05/17/2010

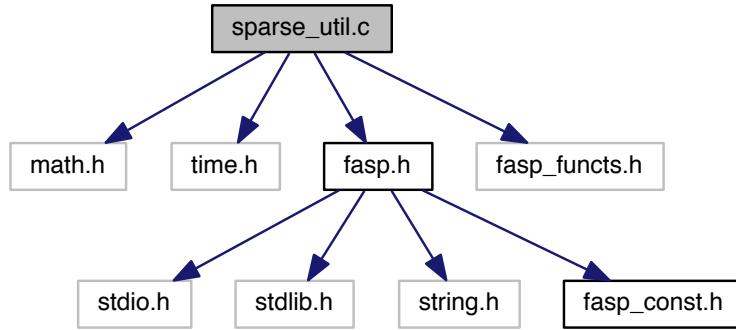
Definition at line 25 of file `sparse_str.c`.

9.93 `sparse_util.c` File Reference

Routines for sparse matrix operations.

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

Include dependency graph for sparse_util.c:



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 $R \cdot A \cdot P$ 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 independent set of a CSR matrix

9.93.1 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

: Modified Xiaozhe Hu 2010-10-18

9.93.2 Function Documentation

9.93.2.1 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

<code>ia</code>	array of row pointers (as usual in CSR)
<code>ja</code>	array of column indices
<code>a</code>	array of entries of the input
<code>na</code>	number of rows of A
<code>ma</code>	number of cols of A
<code>iat</code>	array of row pointers in the result
<code>jat</code>	array of column indices
<code>at</code>	array of entries of the result

Definition at line 272 of file sparse_util.c.

9.93.2.2 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

<code>ia</code>	array of row pointers 1st multiplicand
-----------------	--

<i>ja</i>	array of column indices 1st multiplicand
<i>a</i>	entries of the 1st multiplicand
<i>ib</i>	array of row pointers 2nd multiplicand
<i>jb</i>	array of column indices 2nd multiplicand
<i>b</i>	entries of the 2nd multiplicand
<i>ic</i>	array of row pointers in $c = a * b$
<i>jc</i>	array of column indices in $c = a * b$
<i>c</i>	entries of the result: $c = a * b$
<i>nap</i>	number of rows in the 1st multiplicand
<i>map</i>	number of columns in the 1st multiplicand
<i>mbp</i>	number of columns in the 2nd multiplicand

Modified by Chensong Zhang on 09/11/2012

Definition at line 124 of file sparse_util.c.

9.93.2.3 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 nonzeros.

Parameters

<i>ia</i>	array of row pointers 1st multiplicand
<i>ia</i>	array of row pointers 1st multiplicand
<i>ja</i>	array of column indices 1st multiplicand
<i>ib</i>	array of row pointers 2nd multiplicand
<i>jb</i>	array of column indices 2nd multiplicand
<i>nap</i>	number of rows of A
<i>map</i>	number of cols of A
<i>mbp</i>	number of cols of b
<i>ic</i>	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)
<i>jc</i>	array of column indices in the result $c = a * b$

Modified by Chensong Zhang on 09/11/2012

Definition at line 51 of file sparse_util.c.

9.93.2.4 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 nonzeros.

Parameters

<i>ia</i>	array of row pointers 1st summand
<i>ia</i>	array of row pointers 1st summand
<i>ja</i>	array of column indices 1st summand
<i>ib</i>	array of row pointers 2nd summand

<i>jb</i>	array of column indices 2nd summand
<i>nab</i>	number of rows
<i>mab</i>	number of cols
<i>ic</i>	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)
<i>jc</i>	array of column indices in the result c=a+b

Definition at line 359 of file sparse_util.c.

9.93.2.5 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

<i>ia</i>	array of row pointers 1st summand
<i>ja</i>	array of column indices 1st summand
<i>a</i>	entries of the 1st summand
<i>ib</i>	array of row pointers 2nd summand
<i>jb</i>	array of column indices 2nd summand
<i>b</i>	entries of the 2nd summand
<i>nab</i>	number of rows
<i>mab</i>	number of cols
<i>ic</i>	array of row pointers in c=a+b
<i>jc</i>	array of column indices in c=a+b
<i>c</i>	entries of the result: c=a+b

Definition at line 431 of file sparse_util.c.

9.93.2.6 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

<i>ia</i>	array of row pointers (as usual in CSR)
<i>ja</i>	array of column indices
<i>na</i>	number of rows
<i>ma</i>	number of cols
<i>iat</i>	array of row pointers in the result
<i>jat</i>	array of column indices

Note

For the concrete algorithm, see:

Definition at line 197 of file sparse_util.c.

9.93.2.7 ivector fasp_sparse_MIS (dCSRmat * *A*)

get the maximal independent set of a CSR matrix

Parameters

A	pointer to the matrix
---	-----------------------

Note

: only use the sparsity of A, index starts from 1 (fortran)!!

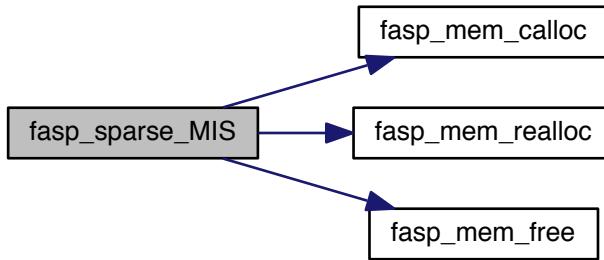
information of A

work space

return

Definition at line 913 of file sparse_util.c.

Here is the call graph for this function:



9.93.2.8 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

<i>ja</i>	:l: array of column indices for A
<i>a</i>	:l: entries of A
<i>ipt</i>	:l: array of row pointers for P
<i>jpt</i>	:l: array of column indices for P
<i>pt</i>	:l: entries of P
<i>nin</i>	:l: number of rows in R
<i>ncin</i>	:l: number of rows in
<i>iac</i>	:O: array of row pointers for P
<i>jac</i>	:O: array of column indices for P
<i>ac</i>	:O: entries of P
<i>idummy</i>	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 791 of file sparse_util.c.

9.93.2.9 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 nonzeros.

Note

:l: is input :O: is output :IO: is both

Parameters

<i>ir</i>	:l: array of row pointers for R
<i>jr</i>	:l: array of column indices for R
<i>ia</i>	:l: array of row pointers for A
<i>ja</i>	:l: array of column indices for A
<i>ip</i>	:l: array of row pointers for P
<i>jp</i>	:l: array of column indices for P
<i>nin</i>	:l: number of rows in R
<i>ncin</i>	:l: number of columns in R
<i>iac</i>	:O: array of row pointers for Ac
<i>jac</i>	:O: array of column indices for Ac
<i>maxrout</i>	:O: the maximum nonzeros per row for R

Note

Computes the sparsity pattern of R*A*P. maxrout is output and is the maximum nonzeros 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.

9.93.2.10 void fasp_sparse_wta_ (INT * *jw*, REAL * *w*, INT * *ia*, INT * *ja*, REAL * *a*, INT * *nwp*, INT * *map*, INT * *iv*, 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

<i>jw</i>	:I: indices such that $w[jw]$ is nonzero
<i>w</i>	:I: the values of w
<i>ia</i>	:I: array of row pointers for A
<i>ja</i>	:I: array of column indices for A
<i>a</i>	:I: entries of A
<i>nwp</i>	:I: number of nonzeros in w (the length of w)
<i>map</i>	:I: number of columns in A
<i>jv</i>	:O: indices such that $v[jv]$ is nonzero
<i>v</i>	:O: the result $v^t = w^t A$
<i>nvp</i>	:I: number of nonzeros in v

Definition at line 651 of file sparse_util.c.

9.93.2.11 void fasp_sparse_wtams_ (INT * *jw*, INT * *ia*, INT * *ja*, INT * *nwp*, INT * *map*, INT * *jv*, INT * *nvp*, INT * *icp*)

Finds the nonzeros 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

<i>jw</i>	:I: indices such that $w[jw]$ is nonzero
<i>ia</i>	:I: array of row pointers for A
<i>ja</i>	:I: array of column indices for A
<i>nwp</i>	:I: number of nonzeros in w (the length of w)
<i>map</i>	:I: number of columns in A
<i>jv</i>	:O: indices such that $v[jv]$ is nonzero
<i>nvp</i>	:I: number of nonzeros in v
<i>icp</i>	:IO: is a working array of length (*map) which on output satisfies $icp[jv[k]-1]=k$; Values of <i>icp</i> [] at positions * other than $(jv[k]-1)$ remain unchanged.

Modified by Chensong Zhang on 09/11/2012

Definition at line 598 of file sparse_util.c.

9.93.2.12 void fasp_sparse_ytx_ (INT * *jy*, REAL * *y*, INT * *ix*, 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

<i>jy</i>	:I: indices such that $y[jy]$ is nonzero
<i>y</i>	:I: is a sparse vector.
<i>nyp</i>	:I: number of nonzeros in y

<i>jx</i>	:l: indices such that $x[jx]$ is nonzero
<i>x</i>	:l: is a sparse vector.
<i>nxp</i>	:l: number of nonzeros in <i>x</i>
<i>icp</i>	???
<i>s</i>	:O: $s = y^T x$.

Definition at line 736 of file sparse_util.c.

9.93.2.13 void fasp_sparse_ytxbig_ (INT *jy, REAL *y, INT *nyp, REAL *x, REAL *s)

Calculates $s = y^T x$. *y*-sparse, *x* - no.

Note

:l: is input :O: is output :IO: is both

Parameters

<i>jy</i>	:l: indices such that $y[jy]$ is nonzero
<i>y</i>	:l: is a sparse vector.
<i>nyp</i>	:l: number of nonzeros in <i>v</i>
<i>x</i>	:l: also a vector assumed to have entry for any $j=jy[i]-1$; for $i=1:nyp$. This means that <i>x</i> here does not have to be sparse.
<i>s</i>	:O: $s = y^T x$.

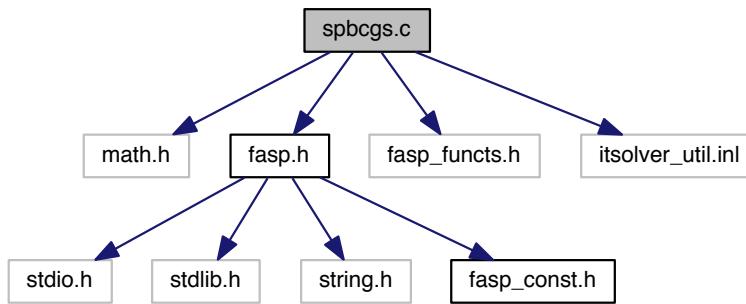
Definition at line 702 of file sparse_util.c.

9.94 spbcgs.c File Reference

Krylov subspace methods – Preconditioned BiCGstab with safe net.

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

Include dependency graph for spbcgs.c:



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** print_level)

Preconditioned BiCGstab method for solving Au=b with safe 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** print_level)

Preconditioned BiCGstab method for solving Au=b with safe net.

- **INT fasp_solver_bdcsr_spbcgs** (**block_dCSRmat** *A, **dvector** *b, **dvector** *u, **precond** *pc, const **REAL** tol, const **INT** MaxIt, const **SHORT** stop_type, const **SHORT** print_level)

Preconditioned BiCGstab method for solving Au=b with safe 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** print_level)

Preconditioned BiCGstab method for solving Au=b with safe net.

9.94.1 Detailed Description

Krylov subspace methods – Preconditioned BiCGstab with safe net.

Abstract algorithm

PBICGStab method to solve $A \cdot x = b$ is to generate $\{x_k\}$ to approximate x

Note: We generate a series of $\{p_k\}$ such that $V_k = \text{span}\{p_1, \dots, p_k\}$.

Step 0. Given A , b , x_0 , M

Step 1. Compute residual $r_0 = b - A \cdot x_0$ and convergence check;

Step 2. Initialization $z_0 = M^{-1} \cdot r_0$, $p_0 = z_0$;

Step 3. Main loop ...

FOR $k = 0$:MaxIt

- get step size $\alpha = f(r_k, z_k, p_k)$;
- update solution: $x_{k+1} = x_k + \alpha \cdot p_k$;
- check whether x is NAN;
- perform stagnation check;
- update residual: $r_{k+1} = r_k - \alpha \cdot (A \cdot p_k)$;
- if $r_{k+1} < r_{\text{best}}$: save x_{k+1} as x_{best} ;
- perform residual check;
- obtain p_{k+1} using $\{p_0, p_1, \dots, p_k\}$;
- prepare for next iteration;
- print the result of k-th iteration; END FOR

Convergence check: $\text{norm}(r)/\text{norm}(b) < \text{tol}$

Stagnation check:

- IF $\text{norm}(\alpha \cdot p_k)/\text{norm}(x_{k+1}) < \text{tol_stag}$

- 1. compute $r = b - A \cdot x_{\{k+1\}}$;
- 2. convergence check;
- 3. IF (not converged & restart_number < Max_Stag_Check) restart;
- END IF

Residual check:

- IF $\text{norm}(r_{\{k+1\}}) / \text{norm}(b) < \text{tol}$
 - 1. compute the real residual $r = b - A \cdot x_{\{k+1\}}$;
 - 2. convergence check;
 - 3. IF (not converged & restart_number < Max_Res_Check) restart;
- END IF

Safe net check:

- IF $r_{\{k+1\}} > r_{\{\text{best}\}}$
 - 1. $x_{\{k+1\}} = x_{\{\text{best}\}}$
- END IF

Note

Refer to Y. Saad 2003 Iterative methods for sparse linear systems (2nd Edition), SIAM
 See [spbcgs.c](#) for a safer version

9.94.2 Function Documentation

9.94.2.1 INT fasp_solver_bdcsr_spbcgs (**block_dCSRmat** * *A*, **dvector** * *b*, **dvector** * *u*, **precond** * *pc*, const **REAL** *tol*, const **INT** *MaxIt*, const **SHORT** *stop_type*, const **SHORT** *print_level*)

Preconditioned BiCGstab method for solving $Au=b$ with safe net.

Parameters

<i>A</i>	Pointer to block_dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector : the right hand side
<i>u</i>	Pointer to dvector : the unknowns
<i>pc</i>	Pointer to the structure of precondition (precond)
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

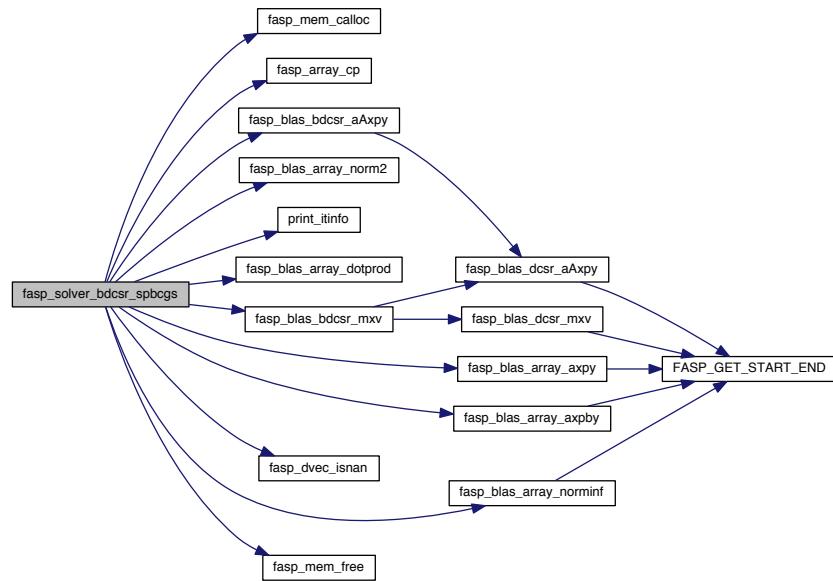
Chensong Zhang

Date

03/31/2013

Definition at line 870 of file spbcgs.c.

Here is the call graph for this function:



9.94.2.2 INT fasp_solver_dbcsr_spbcgs (dBsrmat * A, dvector * b, dvector * u, precond * pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT print_level)

Preconditioned BiCGstab method for solving $Au=b$ with safe net.

Parameters

<i>A</i>	Pointer to dBSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector : the right hand side
<i>u</i>	Pointer to dvector : the unknowns
<i>pc</i>	Pointer to the structure of precondition (precond)
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

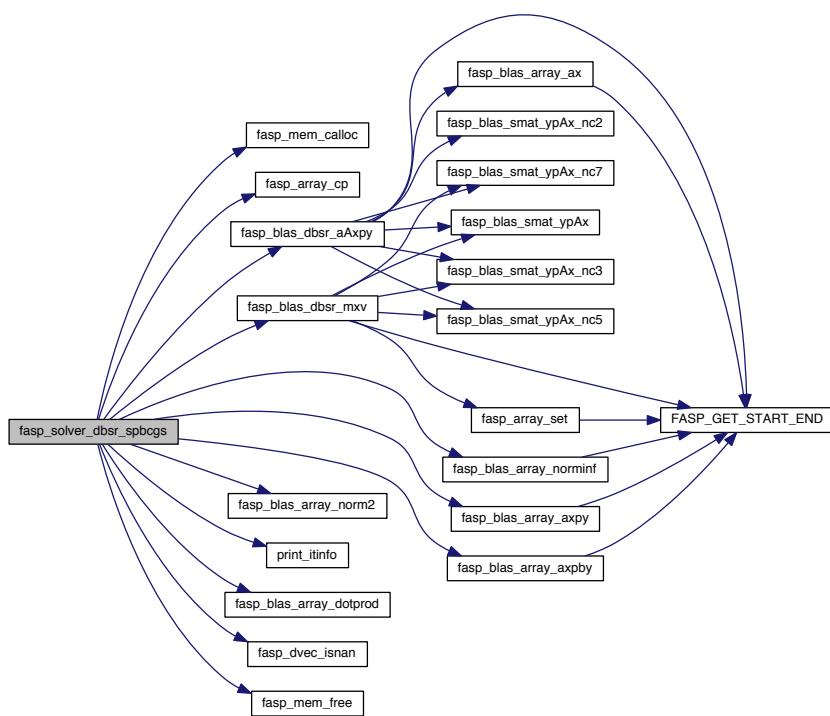
Chensong Zhang

Date

03/31/2013

Definition at line 481 of file spbcgs.c.

Here is the call graph for this function:



9.94.2.3 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 print_level)

Preconditioned BiCGstab method for solving $Au=b$ with safe net.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns

<i>pc</i>	Pointer to the structure of precondition (precond)
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

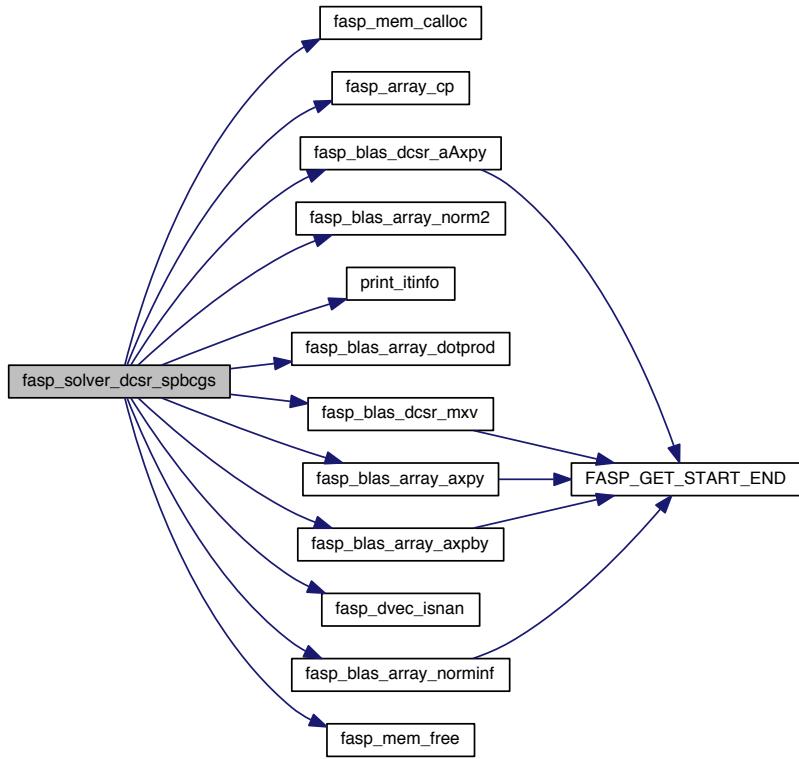
Chensong Zhang

Date

03/31/2013

Definition at line 92 of file spbcgs.c.

Here is the call graph for this function:



9.94.2.4 **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 print_level)**

Preconditioned BiCGstab method for solving Au=b with safe net.

Parameters

<i>A</i>	Pointer to dSTRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to the structure of precondition (precond)
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

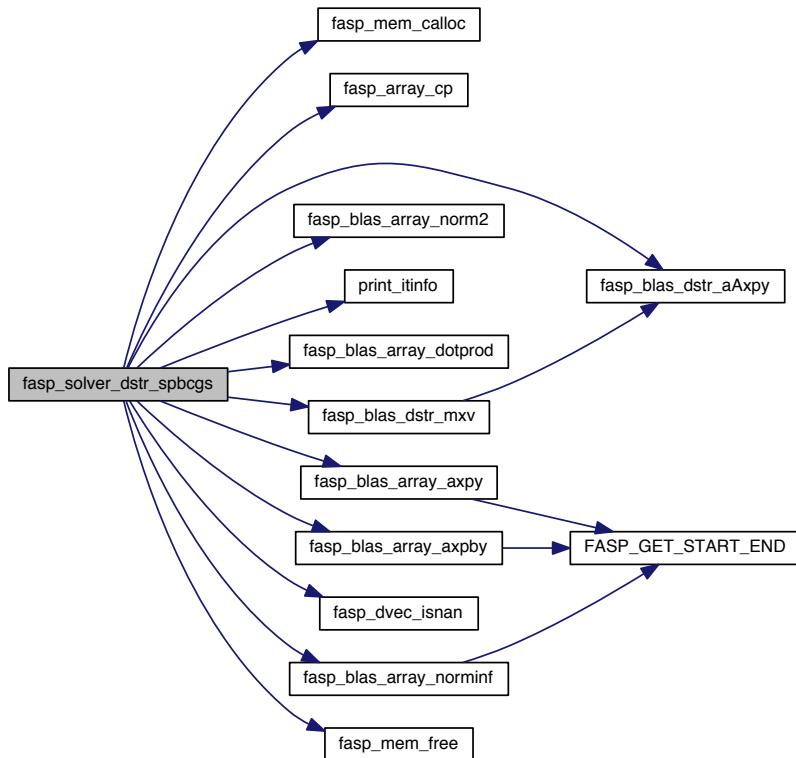
Chensong Zhang

Date

03/31/2013

Definition at line 1259 of file spbcgs.c.

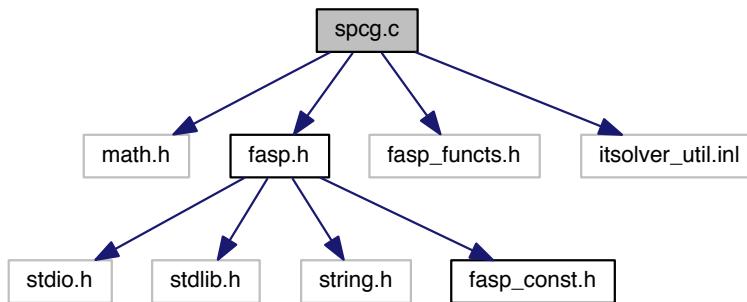
Here is the call graph for this function:



9.95 spcg.c File Reference

Krylov subspace methods – Preconditioned conjugate gradient with safe net.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
Include dependency graph for spcg.c:
```



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 print_level)`
Preconditioned conjugate gradient method for solving $Au=b$ with safe net.
- `INT fasp_solver_bdcsr_spcg (block_dCSRmat *A, dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT print_level)`
Preconditioned conjugate gradient method for solving $Au=b$ with safe 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 print_level)`
Preconditioned conjugate gradient method for solving $Au=b$ with safe net.

9.95.1 Detailed Description

Krylov subspace methods – Preconditioned conjugate gradient with safe 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 $\{p_0, p_1, \dots, p_k\}$;
- prepare for next iteration;
- print the result of k-th iteration; END FOR

Convergence check: $\text{norm}(r)/\text{norm}(b) < \text{tol}$

Stagnation check:

- IF $\text{norm}(\alpha * p_k)/\text{norm}(x_{k+1}) < \text{tol_stag}$
 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 $\text{norm}(r_{k+1})/\text{norm}(b) < \text{tol}$
 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

Safe 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 safe net

9.95.2 Function Documentation

9.95.2.1 INT fasp_solver_bdcsr_spcg (**block_dCSRmat** * *A*, **dvector** * *b*, **dvector** * *u*, **precond** * *pc*, **const REAL** *tol*, **const INT** *MaxIt*, **const SHORT** *stop_type*, **const SHORT** *print_level*)

Preconditioned conjugate gradient method for solving $Au=b$ with safe net.

Parameters

<i>A</i>	Pointer to <code>block_dCSRmat</code> : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to the structure of precondition (precond)
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

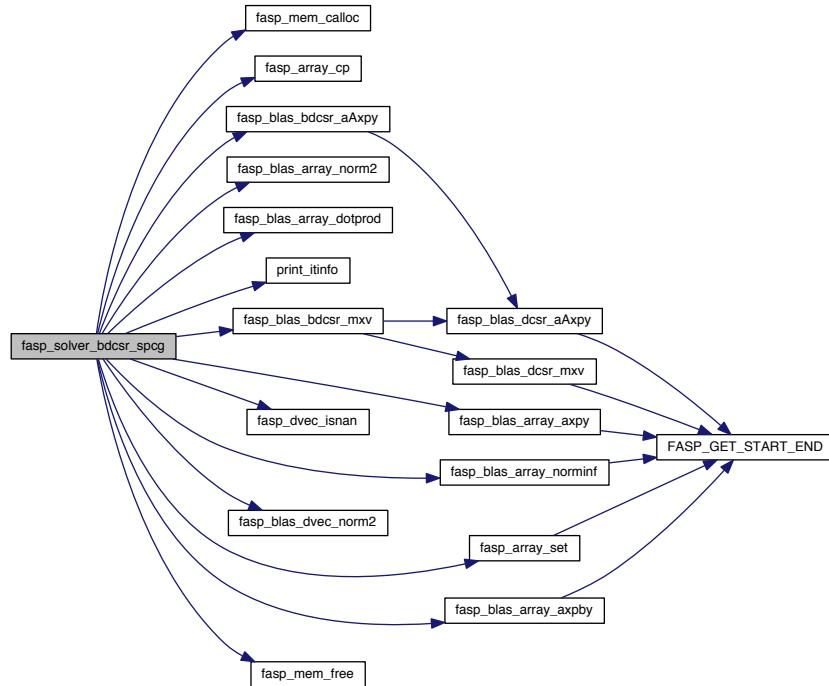
Chensong Zhang

Date

03/28/2013

Definition at line 415 of file spcg.c.

Here is the call graph for this function:



```
9.95.2.2 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 print_level )
```

Preconditioned conjugate gradient method for solving $Au=b$ with safe net.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to the structure of precondition (precond)
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

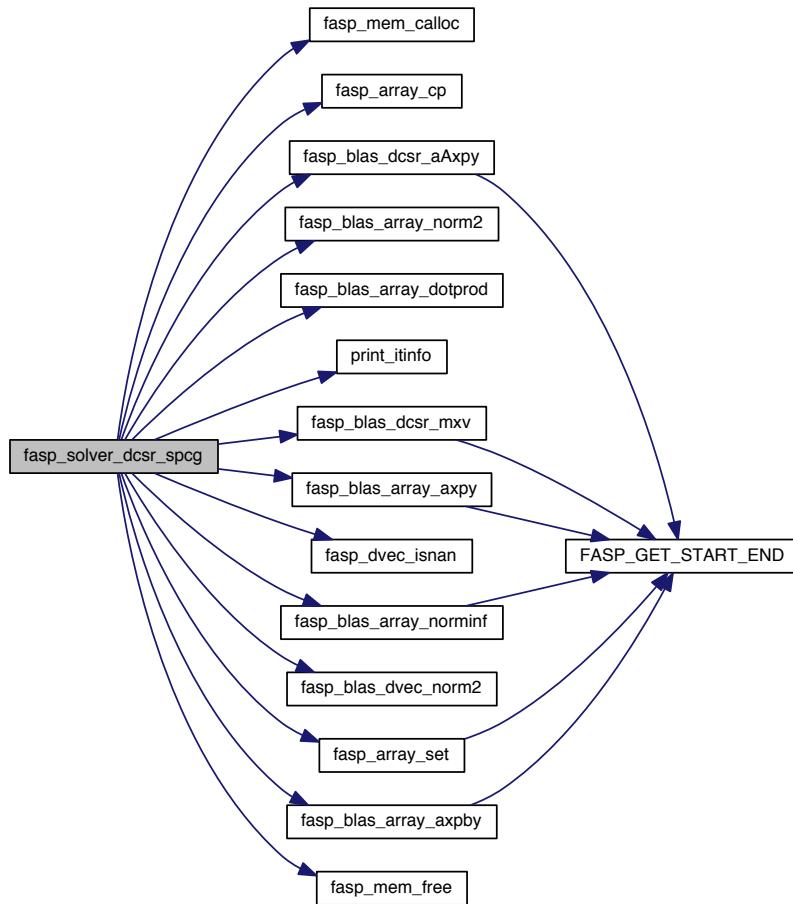
Chensong Zhang

Date

03/28/2013

Definition at line 89 of file spcg.c.

Here is the call graph for this function:



9.95.2.3 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 print_level)

Preconditioned conjugate gradient method for solving $Au=b$ with safe net.

Parameters

<i>A</i>	Pointer to dSTRmat : the coefficient matrix
<i>b</i>	Pointer to dvector : the right hand side
<i>u</i>	Pointer to dvector : the unknowns
<i>MaxIt</i>	Maximal number of iterations
<i>tol</i>	Tolerance for stopping

<i>pc</i>	Pointer to the structure of precondition (precond)
<i>print_level</i>	How much information to print out
<i>stop_type</i>	Stopping criteria type

Returns

Number of iterations if converged, error message otherwise

Author

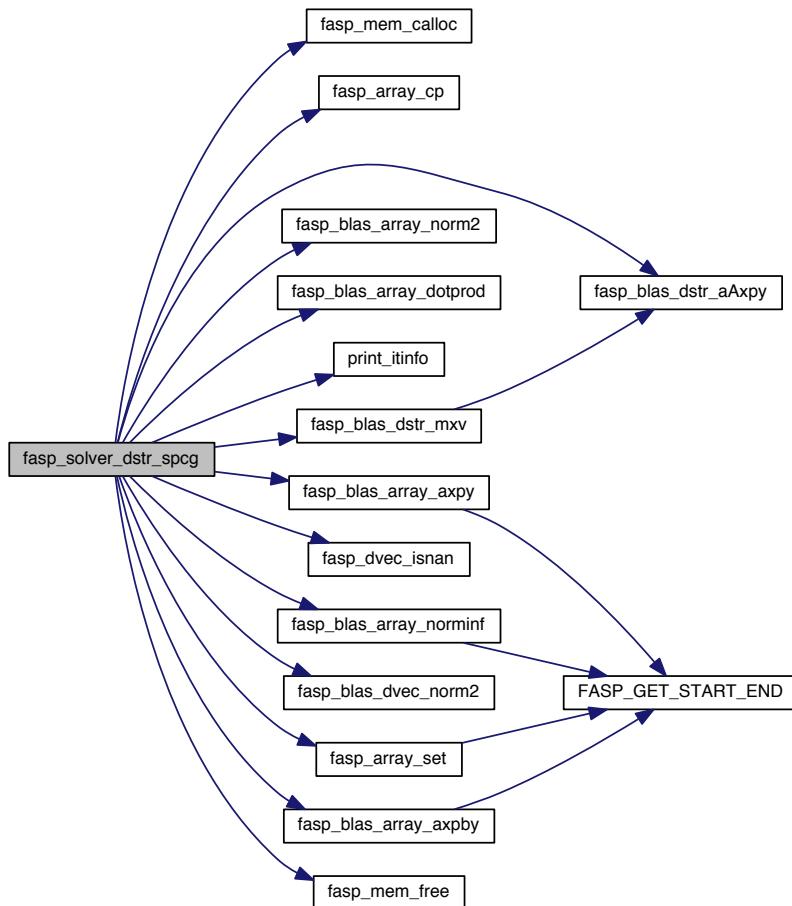
Chensong Zhang

Date

03/28/2013

Definition at line 740 of file spcg.c.

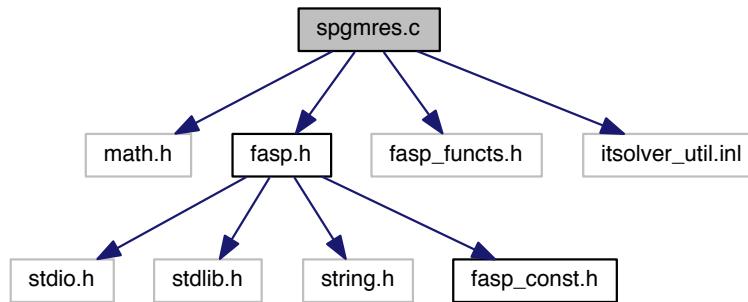
Here is the call graph for this function:



9.96 spgmres.c File Reference

Krylov subspace methods – Preconditioned GMRes with safe net.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
Include dependency graph for spgmres.c:
```



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 print_level)`
Preconditioned GMRES method for solving $Au=b$ with safe-guard.
- `INT fasp_solver_bdcsr_spgmres (block_dCSRmat *A, dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, SHORT restart, const SHORT stop_type, const SHORT print_level)`
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 print_level)`
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 print_level)`
Preconditioned GMRES method for solving $Au=b$ with safe-guard.

9.96.1 Detailed Description

Krylov subspace methods – Preconditioned GMRes with safe 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 safe net

9.96.2 Function Documentation

9.96.2.1 **INT fasp_solver_bdcsr_spgmres (block_dCSRmat * A, dvector * b, dvector * x, precond * pc, const REAL tol, const INT MaxIt, SHORT restart, const SHORT stop_type, const SHORT print_level)**

Preconditioned GMRES method for solving $Au=b$ with safe-guard.

Parameters

<i>A</i>	Pointer to <code>block_dCSRmat</code> : the coefficient matrix
<i>b</i>	Pointer to <code>dvector</code> : the right hand side
<i>x</i>	Pointer to <code>dvector</code> : the unknowns
<i>pc</i>	Pointer to the structure of precondition (precond)
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>restart</i>	Restarting steps
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

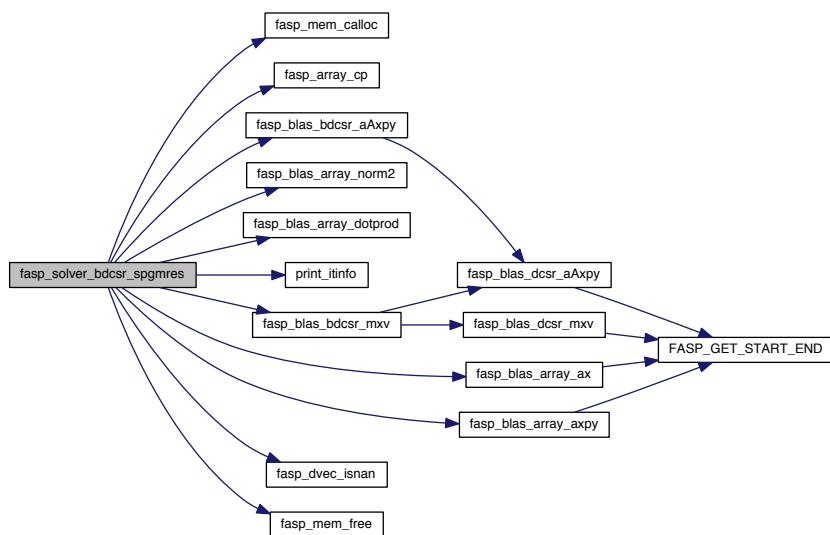
Chensong Zhang

Date

04/05/2013

Definition at line 385 of file `spgmres.c`.

Here is the call graph for this function:



9.96.2.2 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** *print_level*)**

Preconditioned GMRES method for solving $Au=b$ with safe-guard.

Parameters

<i>A</i>	Pointer to dBSRmat : the coefficient matrix
<i>b</i>	Pointer to <i>dvector</i> : the right hand side
<i>x</i>	Pointer to <i>dvector</i> : the unknowns
<i>pc</i>	Pointer to the structure of precondition (precond)
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>restart</i>	Restarting steps
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

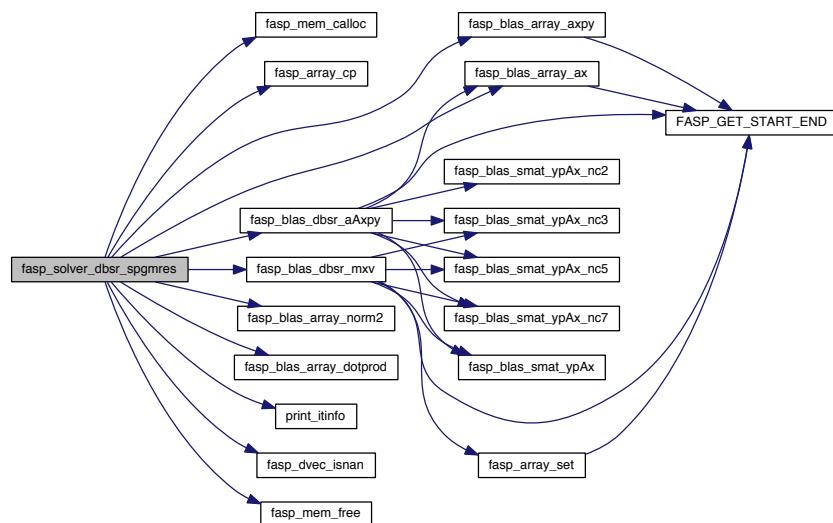
Chensong Zhang

Date

04/05/2013

Definition at line 724 of file spgmres.c.

Here is the call graph for this function:



9.96.2.3 **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 print_level)**

Preconditioned GMRES method for solving $Au=b$ with safe-guard.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>x</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to the structure of precondition (precond)
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>restart</i>	Restarting steps
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

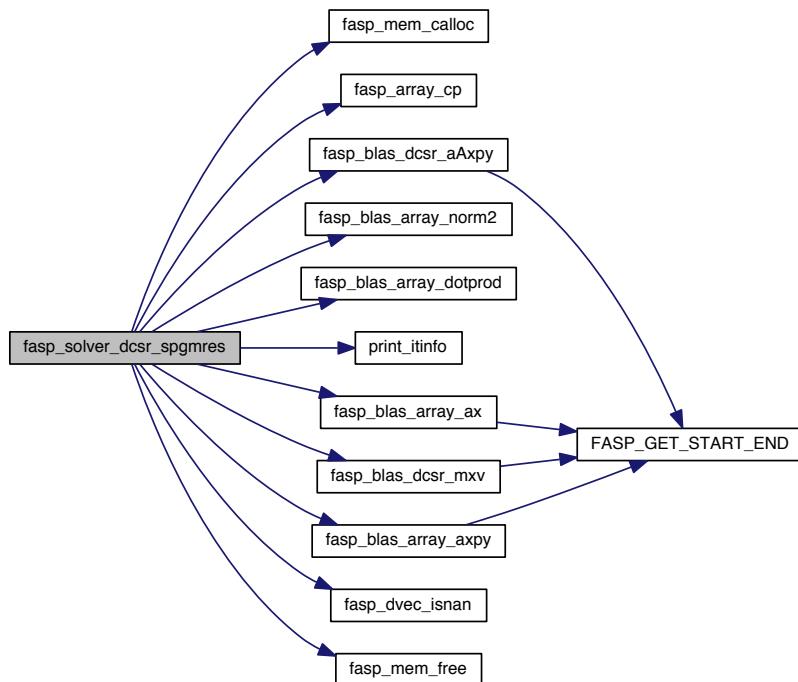
Chensong Zhang

Date

04/05/2013 Modified by Chunsheng Feng on 07/22/2013: Add adapt memory allocate

Definition at line 46 of file spgmres.c.

Here is the call graph for this function:



9.96.2.4 **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 print_level)**

Preconditioned GMRES method for solving $Au=b$ with safe-guard.

Parameters

<i>A</i>	Pointer to dSTRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>x</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to the structure of precondition (precond)
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>restart</i>	Restarting steps
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

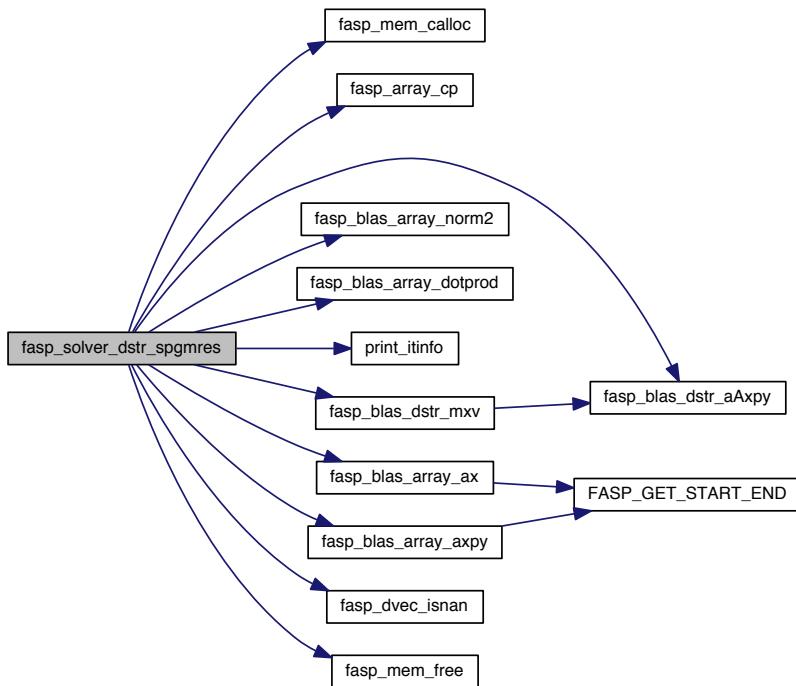
Chensong Zhang

Date

04/05/2013

Definition at line 1063 of file spgmres.c.

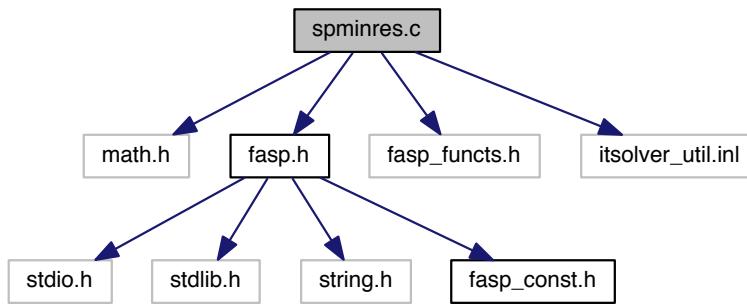
Here is the call graph for this function:



9.97 spminres.c File Reference

Krylov subspace methods – Preconditioned minimal residual with safe net.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "itsolver_util.inl"
Include dependency graph for spminres.c:
```



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 print_level)**
A preconditioned minimal residual (Minres) method for solving $Au=b$ with safe net.
- **INT fasp_solver_bdcsr_spminres (block_dCSRmat *A, dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT print_level)**
A preconditioned minimal residual (Minres) method for solving $Au=b$ with safe 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 print_level)**
A preconditioned minimal residual (Minres) method for solving $Au=b$ with safe net.

9.97.1 Detailed Description

Krylov subspace methods – Preconditioned minimal residual with safe 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 $V_k = \text{span}\{r_0, A*r_0, A^2*r_0, \dots, A^{k-1}*r_0\}$, under some inner product.

For the implementation, we generate a series of $\{p_k\}$ such that $V_k = \text{span}\{p_1, \dots, p_k\}$. 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 $\{p_0, p_1, \dots, p_k\}$;
- prepare for next iteration;
- print the result of k-th iteration; END FOR

Convergence check: $\text{norm}(r)/\text{norm}(b) < \text{tol}$

Stagnation check:

- IF $\text{norm}(\alpha p_k)/\text{norm}(x_{k+1}) < \text{tol_stag}$
 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 $\text{norm}(r_{k+1})/\text{norm}(b) < \text{tol}$
 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

Safe 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 safe net

9.97.2 Function Documentation

9.97.2.1 INT `fasp_solver_bdcsr_spminres(block_dCSRmat * A, dvector * b, dvector * u, precond * pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT print_level)`

A preconditioned minimal residual (Minres) method for solving $Au=b$ with safe net.

Parameters

<i>A</i>	Pointer to <code>block_dCSRmat</code> : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to the structure of precondition (precond)
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

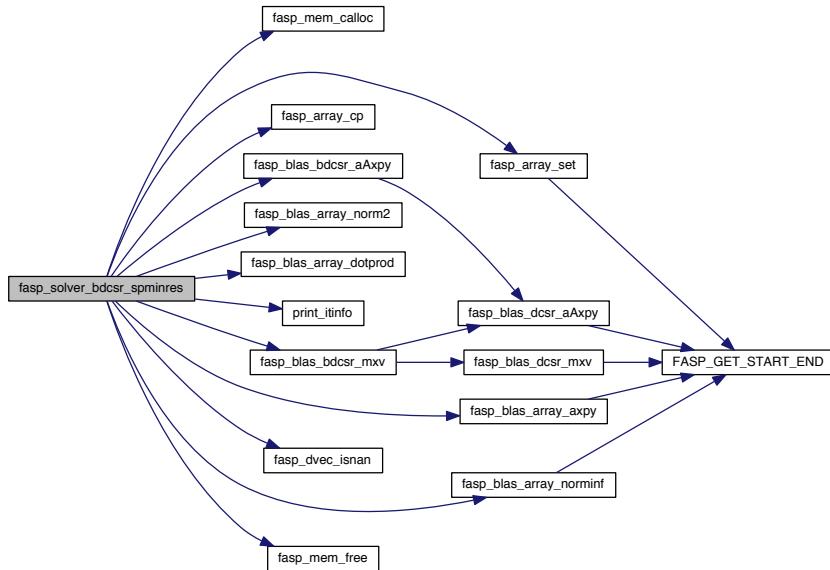
Chensong Zhang

Date

04/09/2013

Definition at line 544 of file spminres.c.

Here is the call graph for this function:



9.97.2.2 INT fasp_solver_dcsr_spminres (dCSRmat * *A*, dvector * *b*, dvector * *u*, precondition * *pc*, const REAL *tol*, const INT *MaxIt*, const SHORT *stop_type*, const SHORT *print_level*)

A preconditioned minimal residual (Minres) method for solving $\mathbf{A}\mathbf{u}=\mathbf{b}$ with safe net.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>u</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to the structure of precondition (precond)
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

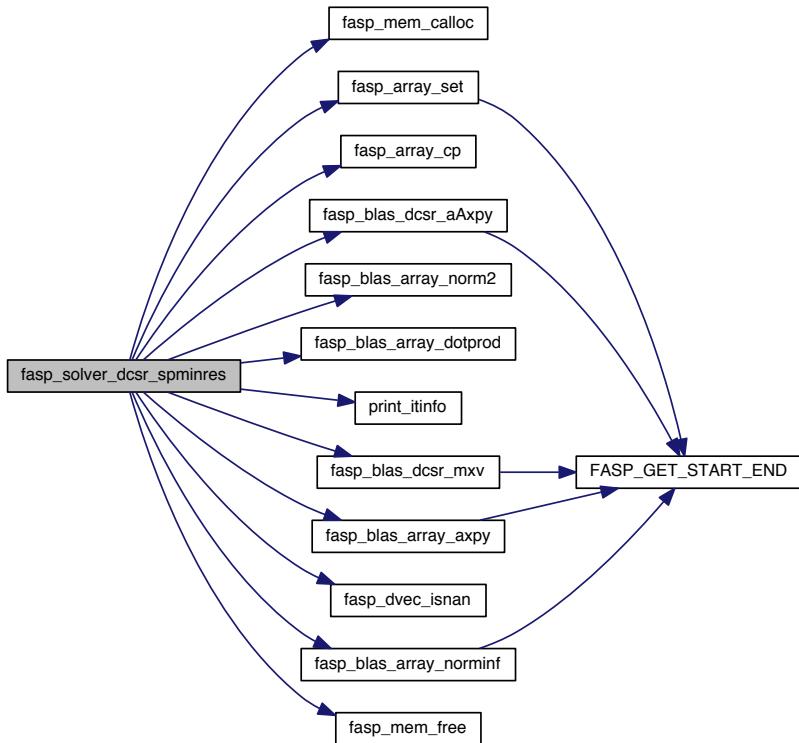
Chensong Zhang

Date

04/09/2013

Definition at line 96 of file spminres.c.

Here is the call graph for this function:



9.97.2.3 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** *print_level*)

A preconditioned minimal residual (Minres) method for solving $Au=b$ with safe net.

Parameters

<i>A</i>	Pointer to dSTRmat : the coefficient matrix
<i>b</i>	Pointer to dvector : the right hand side
<i>u</i>	Pointer to dvector : the unknowns
<i>MaxIt</i>	Maximal number of iterations
<i>tol</i>	Tolerance for stopping
<i>pc</i>	Pointer to the structure of precondition (precond)
<i>print_level</i>	How much information to print out
<i>stop_type</i>	Stopping criteria type

Returns

Number of iterations if converged, error message otherwise

Author

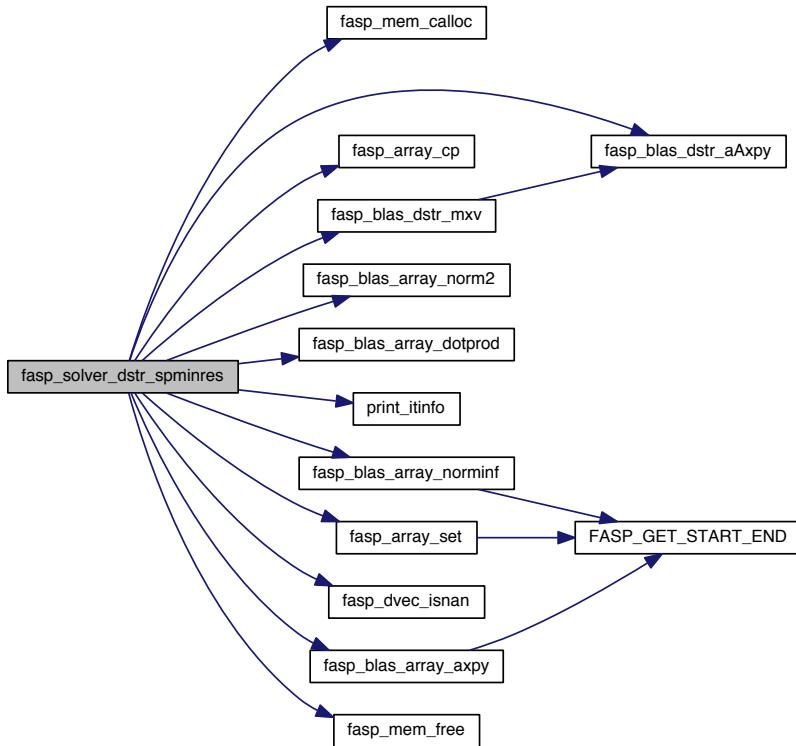
Chensong Zhang

Date

04/09/2013

Definition at line 992 of file spminres.c.

Here is the call graph for this function:

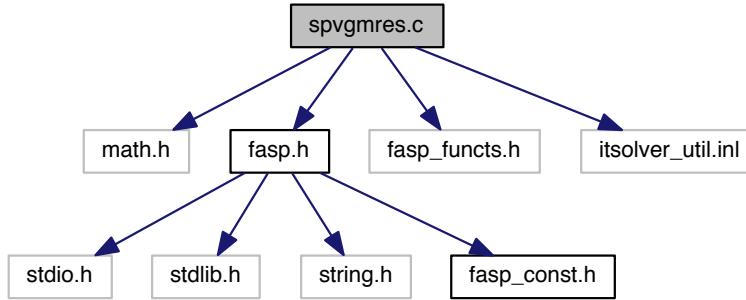


9.98 spvgmres.c File Reference

Krylov subspace methods – Preconditioned variable-restart GMRes with safe net.

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

Include dependency graph for spvgmres.c:



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 print_level)`
Solve "Ax=b" using PGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during the iteration.
- `INT fasp_solver_bdcsr_spvgmres (block_dCSRmat *A, dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, SHORT restart, const SHORT stop_type, const SHORT print_level)`
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 print_level)`
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 print_level)`
Solve "Ax=b" using PGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during the iteration.

9.98.1 Detailed Description

Krylov subspace methods – Preconditioned variable-restart GMRes with safe net.

Note

Refer to A.H. Baker, E.R. Jessup, and Tz.V. Kolev A Simple Strategy for Varying the Restart Parameter in GMR \leftarrow ES(m) Journal of Computational and Applied Mathematics, 230 (2009) pp. 751-761. UCRL-JRNL-235266.
 See [pvgmres.c](#) a version without safe net

9.98.2 Function Documentation

9.98.2.1 INT fasp_solver_bdcsr_spvgmres (**block_dCSRmat** * *A*, **dvector** * *b*, **dvector** * *x*, **precond** * *pc*, const **REAL** *tol*, const **INT** *MaxIt*, **SHORT** *restart*, const **SHORT** *stop_type*, const **SHORT** *print_level*)

Preconditioned GMRES method for solving $Au=b$.

Parameters

<i>A</i>	Pointer to <code>block_dCSRmat</code> : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>x</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to the structure of precondition (precond)
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>restart</i>	Restarting steps
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

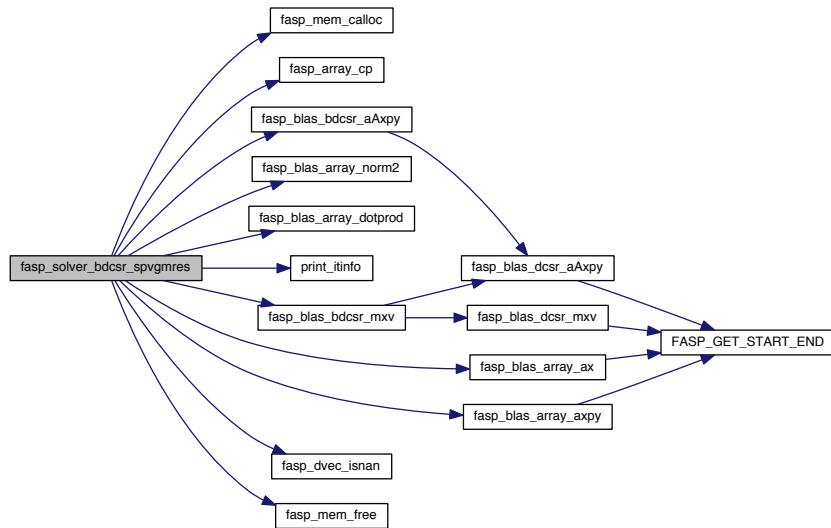
Chensong Zhang

Date

04/06/2013

Definition at line 425 of file spvgmres.c.

Here is the call graph for this function:



9.98.2.2 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** *print_level*)**

Solve "Ax=b" using PGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during the iteration.

Parameters

<i>A</i>	Pointer to dBSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>x</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to the structure of precondition (precond)
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>restart</i>	Restarting steps
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

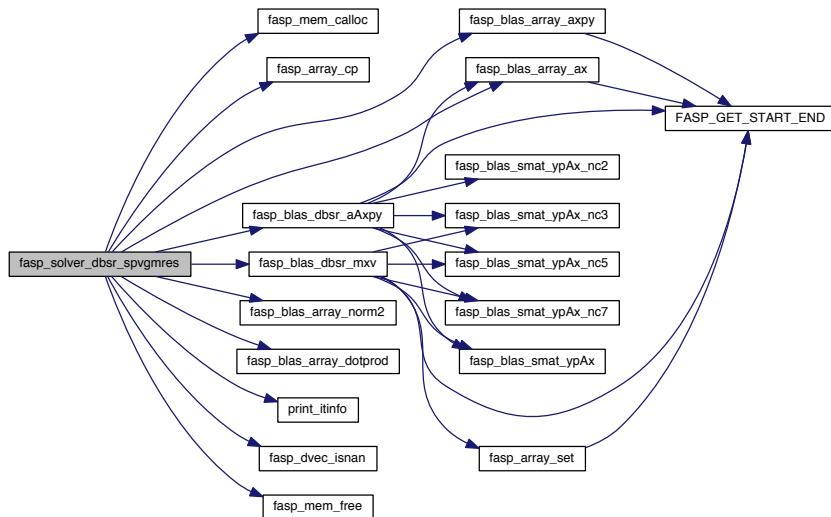
Chensong Zhang

Date

04/06/2013

Definition at line 802 of file spvgmres.c.

Here is the call graph for this function:



9.98.2.3 **INT fasp_solver_dcsr_spvgmres (dCSRmat * *A*, dvector * *b*, dvector * *x*, preconditioner * *pc*, const REAL *tol*, const INT *MaxIt*, SHORT *restart*, const SHORT *stop_type*, const SHORT *print_level*)**

Solve "Ax=b" using PGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during the iteration.

Parameters

<i>A</i>	Pointer to dCSRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>x</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to the structure of precondition (precond)
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>restart</i>	Restarting steps
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

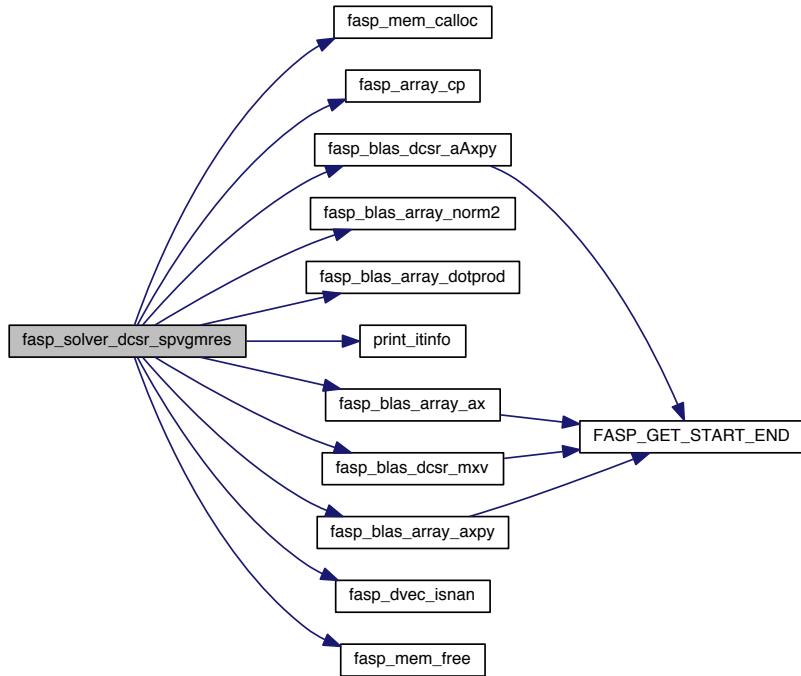
Chensong Zhang

Date

04/06/2013 Modified by Chunsheng Feng on 07/22/2013: Add adapt memory allocate

Definition at line 49 of file spvgmres.c.

Here is the call graph for this function:



9.98.2.4 **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 print_level*)**

Solve "Ax=b" using PGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during the iteration.

Parameters

<i>A</i>	Pointer to dSTRmat : the coefficient matrix
<i>b</i>	Pointer to dvector: the right hand side
<i>x</i>	Pointer to dvector: the unknowns
<i>pc</i>	Pointer to the structure of precondition (precond)
<i>tol</i>	Tolerance for stopping
<i>MaxIt</i>	Maximal number of iterations
<i>restart</i>	Restarting steps
<i>stop_type</i>	Stopping criteria type
<i>print_level</i>	How much information to print out

Returns

Number of iterations if converged, error message otherwise

Author

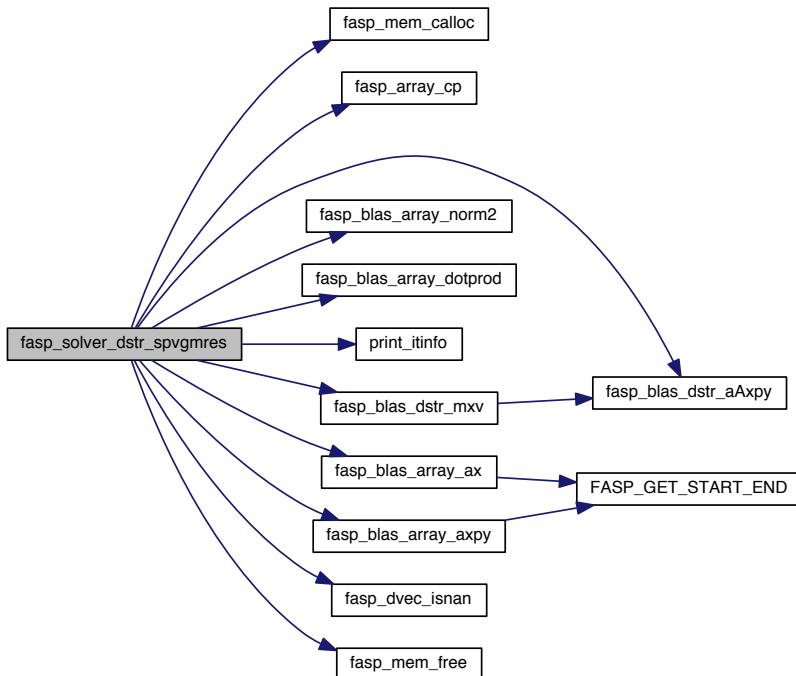
Chensong Zhang

Date

04/06/2013

Definition at line 1179 of file spvgmres.c.

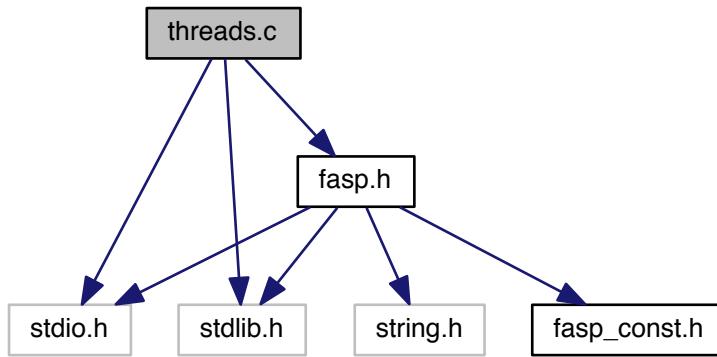
Here is the call graph for this function:



9.99 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"
Include dependency graph for threads.c:
```



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_IGS =0`
- `INT THDs_CPR_gGS =0`

9.99.1 Detailed Description

Get and set number of threads and assign work load for each thread.

9.99.2 Function Documentation

9.99.2.1 void FASP_GET_START_END (INT procid, INT nprocs, INT n, INT * start, INT * end)

Assign Load to each thread.

Parameters

<i>procid</i>	Index of thread
<i>nprocs</i>	Number of threads
<i>n</i>	Total workload
<i>start</i>	Pointer to the begin of each thread in total workload
<i>end</i>	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.

9.99.2.2 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

<i>threads</i>	Total threads of solver
<i>its</i>	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.

9.99.3 Variable Documentation**9.99.3.1 INT THDs_AMG_GS =0**

AMG GS smoothing threads

Definition at line 107 of file threads.c.

9.99.3.2 INT THDs_CPR_gGS =0

global matrix GS smoothing threads

Definition at line 109 of file threads.c.

9.99.3.3 INT THDs_CPR_IGS =0

reservoir GS smoothing threads

Definition at line 108 of file threads.c.

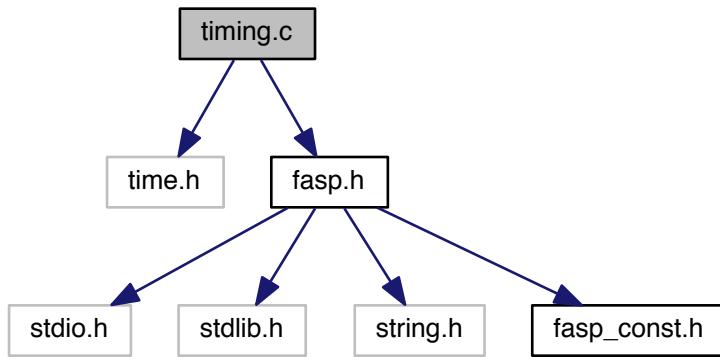
9.100 timing.c File Reference

Timing subroutines.

```
#include <time.h>
```

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

Include dependency graph for timing.c:



Functions

- void [fasp_gettime \(REAL *time\)](#)

Get system time.

9.100.1 Detailed Description

Timing subroutines.

9.100.2 Function Documentation

9.100.2.1 [fasp_gettime \(REAL * time \)](#)

Get system time.

Author

Chunsheng Feng, Zheng Li

Date

11/10/2012

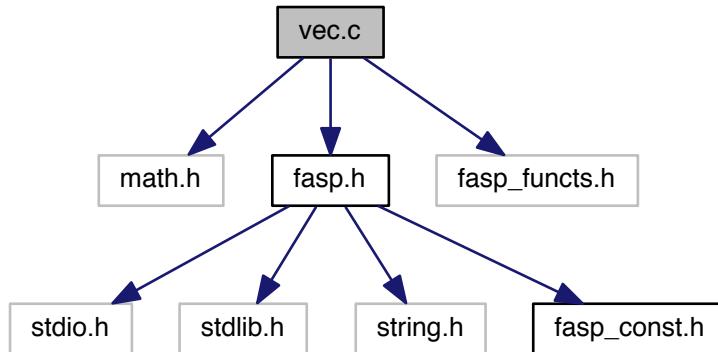
Modified by Chensong Zhang on 09/22/2014: Use CLOCKS_PER_SEC for cross-platform

Definition at line 28 of file timing.c.

9.101 vec.c File Reference

Simple operations for vectors.

```
#include <math.h>
#include "fasp.h"
#include "fasp_functs.h"
Include dependency graph for vec.c:
```



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)**

- void `fasp_dvec_free (dvector *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 $D^{\{-1/2\}}b$.

9.101.1 Detailed Description

Simple operations for vectors.

Note

Every structures should be initialized before usage.

9.101.2 Function Documentation

9.101.2.1 void `fasp_dvec_alloc (const INT m, dvector * u)`

Create dvector data space of REAL type.

Parameters

<i>m</i>	Number of rows
<i>u</i>	Pointer to dvector (OUTPUT)

Author

Chensong Zhang

Date

2010/04/06

Definition at line 99 of file vec.c.

Here is the call graph for this function:

**9.101.2.2 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.

9.101.2.3 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.

Here is the call graph for this function:

**9.101.2.4 void fasp_dvec_free (dvector * u)**

Free vector data space of REAL type.

Parameters

<i>u</i>	Pointer to dvector which needs to be deallocated
----------	--

Author

Chensong Zhang

Date

2010/04/03

Definition at line 139 of file vec.c.

Here is the call graph for this function:

**9.101.2.5 INT fasp_dvec_isnan (dvector * u)**

Check a dvector whether there is NAN.

Parameters

<i>u</i>	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.

9.101.2.6 REAL fasp_dvec_maxdiff (dvector * *x*, dvector * *y*)

Maximal difference of two dvector *x* and *y*.

Parameters

<i>x</i>	Pointer to dvector
<i>y</i>	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.

Here is the call graph for this function:



9.101.2.7 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.

9.101.2.8 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.

9.101.2.9 void fasp_dvec_set (INT n, dvector * x, REAL val)

Initialize dvector $x[i]=val$ for $i=0:n-1$.

Parameters

<i>n</i>	Number of variables
<i>x</i>	Pointer to dvector
<i>val</i>	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.

Here is the call graph for this function:



9.101.2.10 void fasp_dvec_symdiagscale (dvector * *b*, dvector * *diag*)

Symmetric diagonal scaling $D^{-1/2}b$.

Parameters

<i>b</i>	Pointer to dvector
<i>diag</i>	Pointer to dvector: the diagonal entries

Author

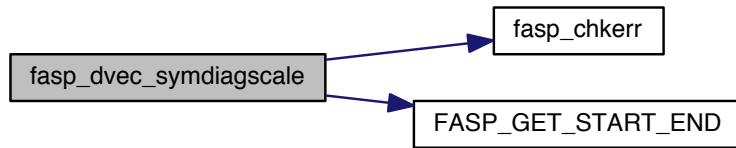
Xiaozhe Hu

Date

01/31/2011

Definition at line 421 of file vec.c.

Here is the call graph for this function:



9.101.2.11 void fasp_ivec_alloc (const INT m , ivec $\ast u$)

Create vector data space of INT type.

Parameters

m	Number of rows
u	Pointer to ivec (OUTPUT)

Author

Chensong Zhang

Date

2010/04/06

Definition at line 119 of file vec.c.

Here is the call graph for this function:



9.101.2.12 ivec \ast fasp_ivec_create (const INT m)

Create vector data space of INT type.

Parameters

<i>m</i>	Number of rows
----------	----------------

Returns

u The new ivector

Author

Chensong Zhang

Date

2010/04/06

Definition at line 78 of file vec.c.

Here is the call graph for this function:

**9.101.2.13 void fasp_ivec_free(ivector * *u*)**

Free vector data space of INT type.

Parameters

<i>u</i>	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.

Here is the call graph for this function:

**9.101.2.14 void fasp_ivec_set(const INT m, ivec * u)**

Set ivec value to be m.

Parameters

<i>m</i>	Integer value of ivec
<i>u</i>	Pointer to ivec (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.

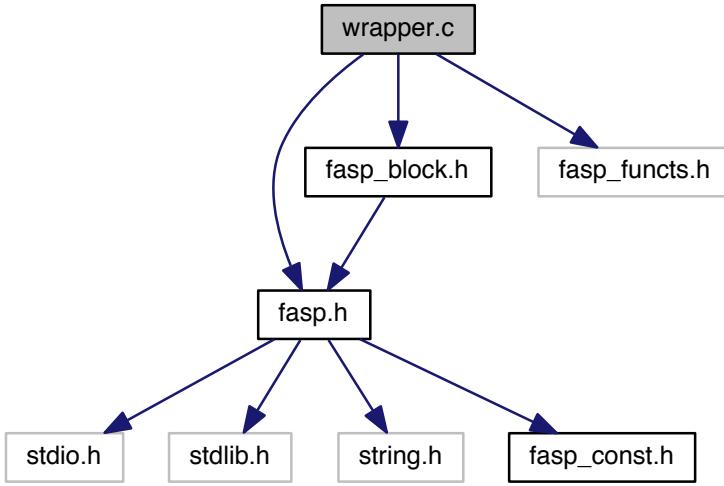
Here is the call graph for this function:



9.102 wrapper.c File Reference

Wrappers for accessing functions by advanced users.

```
#include "fasp.h"
#include "fasp_block.h"
#include "fasp_functs.h"
Include dependency graph for wrapper.c:
```



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)

9.102.1 Detailed Description

Wrappers for accessing functions by advanced users.

TODO: Input variables should not need `fasp.h`!!! –Chensong

9.102.2 Function Documentation

9.102.2.1 void 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

<i>n</i>	Number of cols of A
<i>nnz</i>	Number of nonzeros of A
<i>ia</i>	IA of A in CSR format
<i>ja</i>	JA of A in CSR format
<i>a</i>	VAL of A in CSR format
<i>b</i>	RHS vector
<i>u</i>	Solution vector
<i>tol</i>	Tolerance for iterative solvers
<i>maxit</i>	Max number of iterations
<i>ptrlvl</i>	Print level for iterative solvers

Author

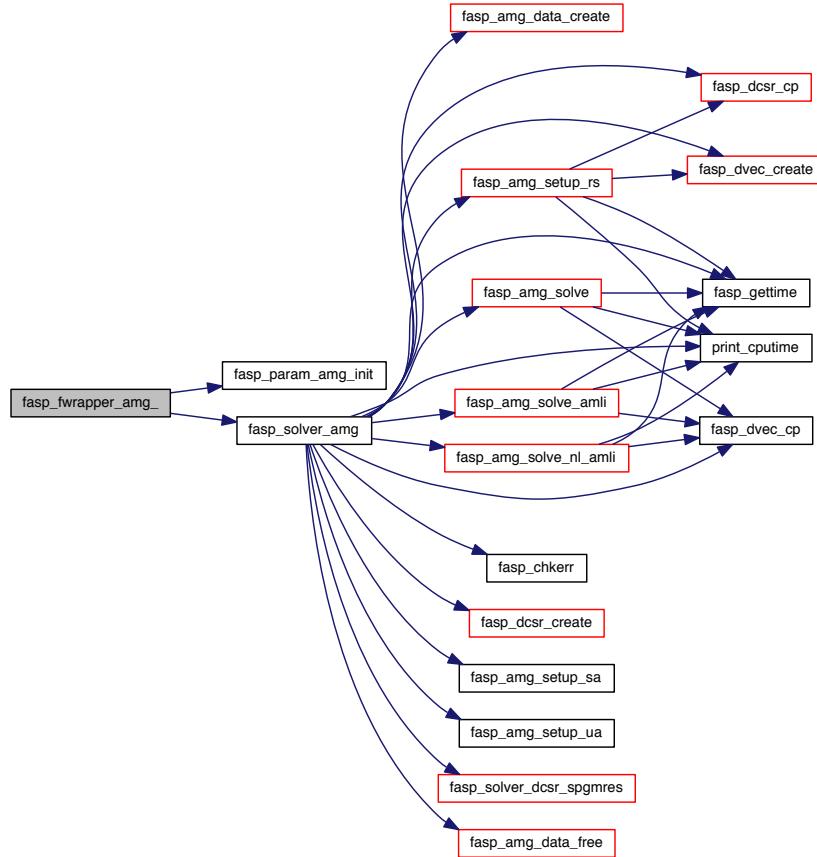
Chensong Zhang

Date

09/16/2010

Definition at line 37 of file wrapper.c.

Here is the call graph for this function:



9.102.2.2 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

<i>n</i>	Number of cols of A
<i>nnz</i>	Number of nonzeros of A
<i>ia</i>	IA of A in CSR format
<i>ja</i>	JA of A in CSR format
<i>a</i>	VAL of A in CSR format
<i>b</i>	RHS vector
<i>u</i>	Solution vector

<i>tol</i>	Tolerance for iterative solvers
<i>maxit</i>	Max number of iterations
<i>ptrlvl</i>	Print level for iterative solvers

Author

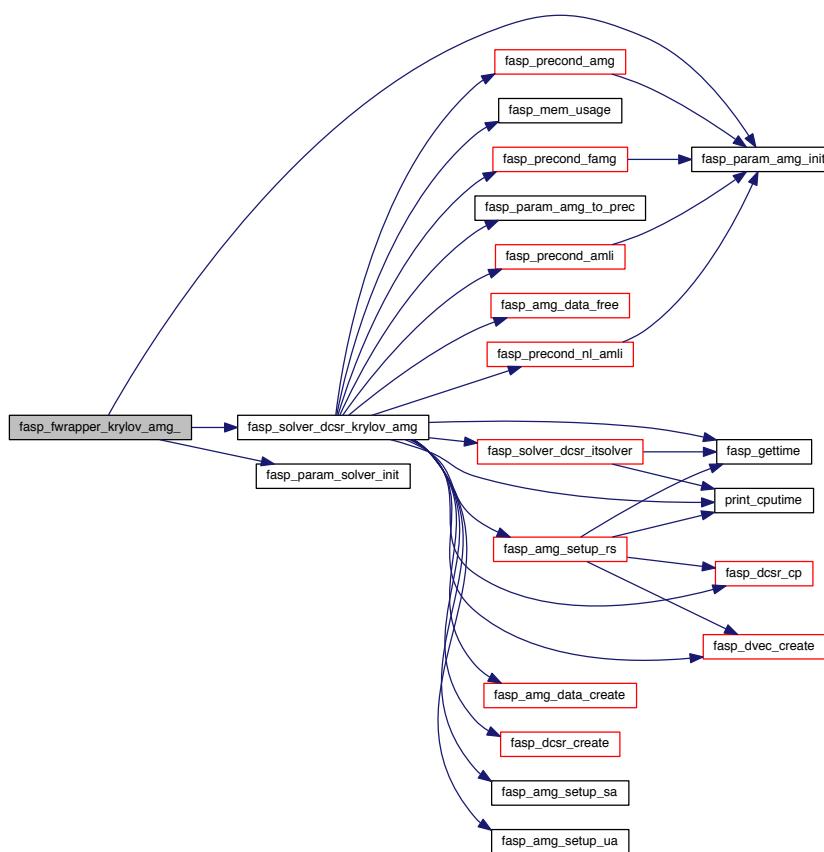
Chensong Zhang

Date

09/16/2010

Definition at line 87 of file wrapper.c.

Here is the call graph for this function:



9.102.2.3 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

<i>n</i>	Number of cols of A
<i>nnz</i>	Number of nonzeros of A
<i>nb</i>	Size of each small block
<i>ia</i>	IA of A in CSR format
<i>ja</i>	JA of A in CSR format
<i>a</i>	VAL of A in CSR format
<i>b</i>	RHS vector
<i>u</i>	Solution vector
<i>tol</i>	Tolerance for iterative solvers
<i>maxit</i>	Max number of iterations
<i>ptrlvl</i>	Print level for iterative solvers

Author

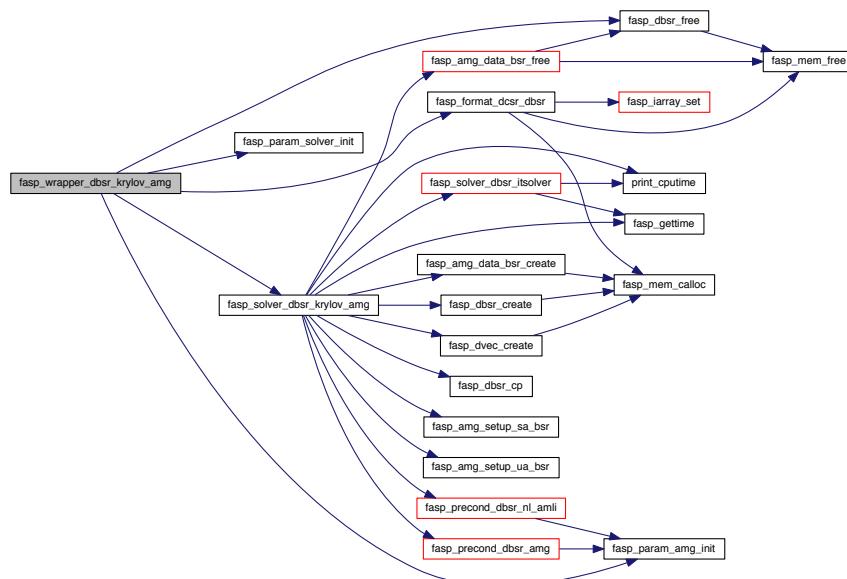
Xiaozhe Hu

Date

03/05/2013

Definition at line 144 of file wrapper.c.

Here is the call graph for this function:



9.102.2.4 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

<i>n</i>	Number of cols of A
<i>nnz</i>	Number of nonzeros of A
<i>nb</i>	Size of each small block
<i>ia</i>	IA of A in COO format
<i>ja</i>	JA of A in COO format
<i>a</i>	VAL of A in COO format
<i>b</i>	RHS vector
<i>u</i>	Solution vector
<i>tol</i>	Tolerance for iterative solvers
<i>maxit</i>	Max number of iterations
<i>ptrlvl</i>	Print level for iterative solvers

Author

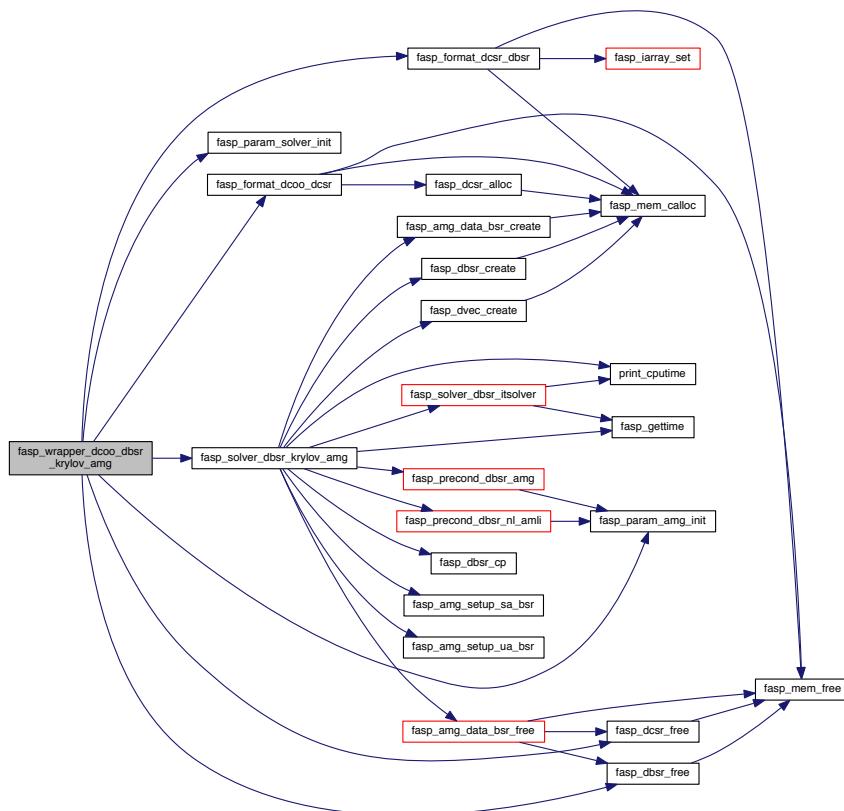
Xiaozhe Hu

Date

03/06/2013

Definition at line 228 of file wrapper.c.

Here is the call graph for this function:



Index

__FASP_HEADER__
fasp.h, 192

A
 precond_FASP_blkoi_data, 69
 precond_sweeping_data, 73

A_diag
 precond_block_data, 54

ABS
 fasp.h, 192

AMG_ILU_levels
 input_param, 43

AMG_Schwarz_levels
 input_param, 45

AMG_aggregation_type
 input_param, 42

AMG_aggressive_level
 input_param, 42

AMG_aggressive_path
 input_param, 43

AMG_amli_degree
 input_param, 43

AMG_coarse_dof
 input_param, 43

AMG_coarse_scaling
 input_param, 43

AMG_coarse_solver
 input_param, 43

AMG_coarsening_type
 input_param, 43

AMG_cycle_type
 input_param, 43

AMG_data, 19

AMG_data_bsr, 20

AMG_interpolation_type
 input_param, 43

AMG_levels
 input_param, 44

AMG_max_aggregation
 input_param, 44

AMG_max_row_sum
 input_param, 44

AMG_maxit
 input_param, 44

AMG_nl_amli_krylov_type
 input_param, 44

 AMG_pair_number
 input_param, 44

 AMG_param, 22

 AMG_polynomial_degree
 input_param, 44

 AMG_postsMOOTH_iter
 input_param, 44

 AMG_presMOOTH_iter
 input_param, 44

 AMG_relaxation
 input_param, 45

 AMG_smooth_filter
 input_param, 45

 AMG_smooth_order
 input_param, 45

 AMG_smoother
 input_param, 45

 AMG_strong_coupled
 input_param, 45

 AMG_strong_threshold
 input_param, 45

 AMG_tentative_smooth
 input_param, 45

 AMG_tol
 input_param, 45

 AMG_truncation_threshold
 input_param, 46

 AMG_type
 input_param, 46

 AMLI_CYCLE
 fasp_const.h, 203

ASCEND
 fasp_const.h, 203

Abcsr
 precond_block_data, 54

Ai
 precond_sweeping_data, 73

amg.c, 77
 fasp_solver_amg, 78

amg_setup_cr.c, 79
 fasp_amg_setup_cr, 80

amg_setup_rs.c, 81
 fasp_amg_setup_rs, 82

amg_setup_sa.c, 84
 fasp_amg_setup_sa, 85

fasp_amg_setup_sa_bsr, 86
 amg_setup_ua.c, 86
 fasp_amg_setup_ua, 87
 fasp_amg_setup_ua_bsr, 87
 amg_solve.c, 88
 fasp_amg_solve, 89
 fasp_amg_solve_amli, 90
 fasp_amg_solve_nl_amli, 92
 fasp_famg_solve, 93
 amgparam
 precond_block_data, 54
 amlirecur.c, 94
 fasp_amg_amli_coef, 95
 fasp_solver_amli, 96
 fasp_solver_nl_amli, 97
 fasp_solver_nl_amli_bsr, 98
 array.c, 101
 fasp_array_cp, 102
 fasp_array_cp_nc3, 102
 fasp_array_cp_nc5, 102
 fasp_array_cp_nc7, 103
 fasp_array_null, 103
 fasp_array_set, 103
 fasp_iarray_cp, 104
 fasp_iarray_set, 104

BIGREAL
 fasp_const.h, 203
 blas_array.c, 105
 fasp_blas_array_ax, 106
 fasp_blas_array_axpby, 107
 fasp_blas_array_axpy, 108
 fasp_blas_array_axpyz, 109
 fasp_blas_array_dotprod, 109
 fasp_blas_array_norm1, 110
 fasp_blas_array_norm2, 110
 fasp_blas_array_norminf, 111
 blas_bcsr.c, 112
 fasp_blas_bdbsr_aAxpy, 112
 fasp_blas_bdbsr_mxv, 114
 fasp_blas_bdcsr_aAxpy, 115
 fasp_blas_bdcsr_mxv, 115
 blas_bsr.c, 116
 fasp_blas_dbsr_aAxpby, 118
 fasp_blas_dbsr_aAxpy, 119
 fasp_blas_dbsr_aAxpy_agg, 120
 fasp_blas_dbsr_axm, 121
 fasp_blas_dbsr_mxmm, 122
 fasp_blas_dbsr_mxv, 123
 fasp_blas_dbsr_mxv_agg, 124
 fasp_blas_dbsr_rap, 125
 fasp_blas_dbsr_rap1, 126
 fasp_blas_dbsr_rap_agg, 127
 blas_csr.c, 128

 fasp_blas_dCSR_aAxpy, 129
 fasp_blas_dCSR_aAxpy_agg, 130
 fasp_blas_dCSR_add, 131
 fasp_blas_dCSR_axm, 131
 fasp_blas_dCSR_mxmm, 132
 fasp_blas_dCSR_mxv, 133
 fasp_blas_dCSR_mxv_agg, 133
 fasp_blas_dCSR_ptap, 135
 fasp_blas_dCSR_rap, 136
 fasp_blas_dCSR_rap4, 137
 fasp_blas_dCSR_rap_agg, 138
 fasp_blas_dCSR_rap_agg1, 139
 fasp_blas_dCSR_vmv, 140
 blas_CSRl.c, 140
 fasp_blas_dCSRl_mxv, 141
 blas_smat.c, 142
 fasp_blas_array_axpy_nc2, 144
 fasp_blas_array_axpy_nc3, 144
 fasp_blas_array_axpy_nc5, 144
 fasp_blas_array_axpy_nc7, 145
 fasp_blas_array_axpyz_nc2, 145
 fasp_blas_array_axpyz_nc3, 146
 fasp_blas_array_axpyz_nc5, 146
 fasp_blas_array_axpyz_nc7, 146
 fasp_blas_smat_aAxpby, 148
 fasp_blas_smat_add, 148
 fasp_blas_smat_axm, 149
 fasp_blas_smat_mul, 149
 fasp_blas_smat_mul_nc2, 150
 fasp_blas_smat_mul_nc3, 150
 fasp_blas_smat_mul_nc5, 151
 fasp_blas_smat_mul_nc7, 151
 fasp_blas_smat_mxv, 151
 fasp_blas_smat_mxv_nc2, 153
 fasp_blas_smat_mxv_nc3, 154
 fasp_blas_smat_mxv_nc5, 154
 fasp_blas_smat_mxv_nc7, 154
 fasp_blas_smat_ymAx, 155
 fasp_blas_smat_ymAx_nc2, 155
 fasp_blas_smat_ymAx_nc3, 156
 fasp_blas_smat_ymAx_nc5, 156
 fasp_blas_smat_ymAx_nc7, 156
 fasp_blas_smat_ymAx_ns, 157
 fasp_blas_smat_ymAx_ns2, 157
 fasp_blas_smat_ymAx_ns3, 158
 fasp_blas_smat_ymAx_ns5, 158
 fasp_blas_smat_ymAx_ns7, 159
 fasp_blas_smat_ypAx, 159
 fasp_blas_smat_ypAx_nc2, 160
 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

faspblas_dstr_mxv, 164
fasp_dstr_diagscale, 164
blas_vec.c, 165
 faspblas_dvec_axpy, 166
 faspblas_dvec_axpyz, 167
 faspblas_dvec_dotprod, 168
 faspblas_dvec_norm1, 168
 faspblas_dvec_norm2, 169
 faspblas_dvec_norminf, 169
 faspblas_dvec_relerr, 170
block_BSR, 25
 fasp_block.h, 199
block_Reservoir, 29
 fasp_block.h, 199
block_dCSRmat, 25
 fasp_block.h, 199
block_dvector, 26
 fasp_block.h, 199
block_iCSRmat, 27
 fasp_block.h, 199
block_ivector, 28
 fasp_block.h, 199

CF_ORDER
 fasp_const.h, 203
CGPT
 fasp_const.h, 203
CLASSIC_AMG
 fasp_const.h, 203
COARSE_AC
 fasp_const.h, 204
COARSE_CR
 fasp_const.h, 204
COARSE_MIS
 fasp_const.h, 204
COARSE_RS
 fasp_const.h, 204
CPFIRST
 fasp_const.h, 204
checkmat.c, 171
 fasp_check_dCSRmat, 172
 fasp_check_diagdom, 173
 fasp_check_diagpos, 174
 fasp_check_diagzero, 175
 fasp_check_iCSRmat, 175
 fasp_check_symm, 175
coarsening_cr.c, 176
 fasp_amg_coarsening_cr, 177
coarsening_rs.c, 178
 fasp_amg_coarsening_rs, 179
convert.c, 180
 endian_convert_int, 181
 endian_convert_real, 182
 fasp_aux_bbyteTodouble, 182

fasp_aux_change_endian4, 183
fasp_aux_change_endian8, 183
count
 fasp.h, 196
dBSRmat, 30
 fasp_block.h, 199
 JA, 31
 val, 31
dCOOmat, 32
 fasp.h, 194
dCSRLmat, 32
 fasp.h, 194
dCSRmat, 33
 fasp.h, 194
dCSRmat2SAMGInput
 interface_samg.c, 266
DESCEND
 fasp_const.h, 204
DIAGONAL_PREF
 fasp.h, 192
DL_MALLOC
 fasp.h, 192
dSTRmat, 34
 fasp.h, 195
ddenmat, 34
 fasp.h, 195
diag
 precond_block_reservoir_data, 57
diaginv
 precond_FASP_blkoil_data, 69
 precond_block_reservoir_data, 57
diaginv_S
 precond_FASP_blkoil_data, 69
diaginv_noscale
 precond_FASP_blkoil_data, 69
diaginvS
 precond_block_reservoir_data, 57
dlength
 io.c, 308
doxygen.h, 184
dvector, 35
 fasp.h, 195
dvector2SAMGInput
 interface_samg.c, 267

e
 grid2d, 36
ERROR_ALLOC_MEM
 fasp_const.h, 204
ERROR_AMG_COARSE_TYPE
 fasp_const.h, 204
ERROR_AMG_COARSEING
 fasp_const.h, 205
ERROR_AMG_INTERP_TYPE

fasp_const.h, 205
ERROR_AMG_SMOOTH_TYPE
 fasp_const.h, 205
ERROR_DATA_STRUCTURE
 fasp_const.h, 205
ERROR_DATA_ZERODIAG
 fasp_const.h, 205
ERROR_DUMMY_VAR
 fasp_const.h, 205
ERROR_INPUT_PAR
 fasp_const.h, 205
ERROR_LIC_TYPE
 fasp_const.h, 205
ERROR_MAT_SIZE
 fasp_const.h, 205
ERROR_MISC
 fasp_const.h, 206
ERROR_NUM_BLOCKS
 fasp_const.h, 206
ERROR_OPEN_FILE
 fasp_const.h, 206
ERROR_QUAD_DIM
 fasp_const.h, 206
ERROR_QUAD_TYPE
 fasp_const.h, 206
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, 207
ERROR_SOLVER_PRECTYPE
 fasp_const.h, 207
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, 36
ediri
 grid2d, 37
efather
 grid2d, 37
eigen.c, 184
 fasp_dcsr_eig, 185
endian_convert_int
 convert.c, 181
endian_convert_real
 convert.c, 182
FALSE
 fasp_const.h, 207
FASP_GET_START_END
 threads.c, 646
FASP_GSRB
 fasp.h, 192
FASP_SUCCESS
 fasp_const.h, 208
FASP_USE_ILU
 fasp.h, 192
FGPT
 fasp_const.h, 208
FPPFIRST
 fasp_const.h, 208
factor.f, 186
famg.c, 186
 fasp_solver_famg, 187
fasp.h, 188
 __FASP_HEADER__, 192
ABS, 192
count, 196
dCOOmat, 194
DCSRLmat, 194
dCSRmat, 194
DIAGONAL_PREF, 192
DLMALLOC, 192
dSTRmat, 195
ddenmat, 195
dvector, 195
FASP_GSRB, 192
FASP_USE_ILU, 192
GE, 192
GT, 192
 grid2d, 195
iCOOmat, 195
iCSRmat, 195
IMAP, 196
INT, 193
ISNAN, 193
idenmat, 195
ivector, 195
LE, 193
LONG, 193
LONGLONG, 193
LS, 193
LinkList, 195

ListElement, 195
MAX, 193
MAXIMAP, 196
MIN, 193
NEDMALLOC, 193
nx_rb, 196
ny_rb, 196
nz_rb, 196
PRT_INT, 194
PRT_REAL, 194
pcgrid2d, 195
pgrid2d, 196
REAL, 194
RS_C1, 194
SHORT, 194
total_alloc_count, 196
total_alloc_mem, 196
fasp_BinarySearch
 ordering.c, 373
fasp_Schwarz_data_free
 init.c, 261
fasp_Schwarz_get_block_matrix
 schwarz_setup.c, 495
fasp_Schwarz_setup
 schwarz_setup.c, 496
fasp_amg_amli_coef
 amlirecur.c, 95
fasp_amg_coarsening_cr
 coarsening_cr.c, 177
fasp_amg_coarsening_rs
 coarsening_rs.c, 179
fasp_amg_data_bsr_create
 init.c, 255
fasp_amg_data_bsr_free
 init.c, 255
fasp_amg_data_create
 init.c, 256
fasp_amg_data_free
 init.c, 257
fasp_amg_interp
 interpolation.c, 272
fasp_amg_interp1
 interpolation.c, 272
fasp_amg_interp_em
 interpolation_em.c, 275
fasp_amg_interp_trunc
 interpolation.c, 273
fasp_amg_setup_cr
 amg_setup_cr.c, 80
fasp_amg_setup_rs
 amg_setup_rs.c, 82
fasp_amg_setup_sa
 amg_setup_sa.c, 85
fasp_amg_setup_sa_bsr
 amg_setup_sa.c, 86
fasp_amg_setup_ua
 amg_setup_ua.c, 87
fasp_amg_setup_ua_bsr
 amg_setup_ua.c, 87
fasp_amg_solve
 amg_solve.c, 89
fasp_amg_solve_amli
 amg_solve.c, 90
fasp_amg_solve_nl_amli
 amg_solve.c, 92
fasp_array_cp
 array.c, 102
fasp_array_cp_nc3
 array.c, 102
fasp_array_cp_nc5
 array.c, 102
fasp_array_cp_nc7
 array.c, 103
fasp_array_null
 array.c, 103
fasp_array_set
 array.c, 103
fasp_aux_bbyteTodouble
 convert.c, 182
fasp_aux_change_endian4
 convert.c, 183
fasp_aux_change_endian8
 convert.c, 183
fasp_aux_dQuickSort
 ordering.c, 370
fasp_aux_dQuickSortIndex
 ordering.c, 370
fasp_aux_givens
 givens.c, 230
fasp_aux_iQuickSort
 ordering.c, 370
fasp_aux_iQuickSortIndex
 ordering.c, 371
fasp_aux_merge
 ordering.c, 371
fasp_aux_msort
 ordering.c, 372
fasp_aux_unique
 ordering.c, 373
fasp_bdcsr_free
 sparse_block.c, 556
fasp blas array_ax
 blas_array.c, 106
fasp blas array_axpby
 blas_array.c, 107
fasp blas array_axpy
 blas_array.c, 108
fasp blas array_axpy_nc2

blas_smat.c, 144
 faspblas_array_axpy_nc3
 blas_smat.c, 144
 faspblas_array_axpy_nc5
 blas_smat.c, 144
 faspblas_array_axpy_nc7
 blas_smat.c, 145
 faspblas_array_axpyz
 blas_array.c, 109
 faspblas_array_axpyz_nc2
 blas_smat.c, 145
 faspblas_array_axpyz_nc3
 blas_smat.c, 146
 faspblas_array_axpyz_nc5
 blas_smat.c, 146
 faspblas_array_axpyz_nc7
 blas_smat.c, 146
 faspblas_array_dotprod
 blas_array.c, 109
 faspblas_array_norm1
 blas_array.c, 110
 faspblas_array_norm2
 blas_array.c, 110
 faspblas_array_norminf
 blas_array.c, 111
 faspblas_bdbsr_aAxpy
 blas_bcsr.c, 112
 faspblas_bdbsr_mxv
 blas_bcsr.c, 114
 faspblas_bdcsr_aAxpy
 blas_bcsr.c, 115
 faspblas_bdcsr_mxv
 blas_bcsr.c, 115
 faspblas_dbsr_aAxpby
 blas_bsr.c, 118
 faspblas_dbsr_aAxpy
 blas_bsr.c, 119
 faspblas_dbsr_aAxpy_agg
 blas_bsr.c, 120
 faspblas_dbsr_axm
 blas_bsr.c, 121
 faspblas_dbsr_mxm
 blas_bsr.c, 122
 faspblas_dbsr_mxv
 blas_bsr.c, 123
 faspblas_dbsr_mxv_agg
 blas_bsr.c, 124
 faspblas_dbsr_rap
 blas_bsr.c, 125
 faspblas_dbsr_rap1
 blas_bsr.c, 126
 faspblas_dbsr_rap_agg
 blas_bsr.c, 127
 faspblas_dcsr_aAxpy
 blas_csr.c, 129
 faspblas_dcsr_aAxpy_agg
 blas_csr.c, 130
 faspblas_dcsr_add
 blas_csr.c, 131
 faspblas_dcsr_axm
 blas_csr.c, 131
 faspblas_dcsr_mxm
 blas_csr.c, 132
 faspblas_dcsr_mxv
 blas_csr.c, 133
 faspblas_dcsr_mxv_agg
 blas_csr.c, 133
 faspblas_dcsr_ptap
 blas_csr.c, 135
 faspblas_dcsr_rap
 blas_csr.c, 136
 faspblas_dcsr_rap2
 rap.c, 490
 faspblas_dcsr_rap4
 blas_csr.c, 137
 faspblas_dcsr_rap_agg
 blas_csr.c, 138
 faspblas_dcsr_rap_agg1
 blas_csr.c, 139
 faspblas_dcsr_vmv
 blas_csr.c, 140
 faspblas_dcsrl_mxv
 blas_csr.c, 141
 faspblas_dstr_aAxpy
 blas_str.c, 163
 faspblas_dstr_mxv
 blas_str.c, 164
 faspblas_dvec_axpy
 blas_vec.c, 166
 faspblas_dvec_axpyz
 blas_vec.c, 167
 faspblas_dvec_dotprod
 blas_vec.c, 168
 faspblas_dvec_norm1
 blas_vec.c, 168
 faspblas_dvec_norm2
 blas_vec.c, 169
 faspblas_dvec_norminf
 blas_vec.c, 169
 faspblas_dvec_relerr
 blas_vec.c, 170
 faspblas_smat_Linfinity
 smat.c, 501
 faspblas_smat_aAxpby
 blas_smat.c, 148
 faspblas_smat_add
 blas_smat.c, 148
 faspblas_smat_axm

blas_smat.c, 149
fasp_blas_smat_inv
smat.c, 499
fasp_blas_smat_inv_nc2
smat.c, 499
fasp_blas_smat_inv_nc3
smat.c, 500
fasp_blas_smat_inv_nc4
smat.c, 500
fasp_blas_smat_inv_nc5
smat.c, 500
fasp_blas_smat_inv_nc7
smat.c, 501
fasp_blas_smat_mul
blas_smat.c, 149
fasp_blas_smat_mul_nc2
blas_smat.c, 150
fasp_blas_smat_mul_nc3
blas_smat.c, 150
fasp_blas_smat_mul_nc5
blas_smat.c, 151
fasp_blas_smat_mul_nc7
blas_smat.c, 151
fasp_blas_smat_mxv
blas_smat.c, 151
fasp_blas_smat_mxv_nc2
blas_smat.c, 153
fasp_blas_smat_mxv_nc3
blas_smat.c, 154
fasp_blas_smat_mxv_nc5
blas_smat.c, 154
fasp_blas_smat_mxv_nc7
blas_smat.c, 154
fasp_blas_smat_ymAx
blas_smat.c, 155
fasp_blas_smat_ymAx_nc2
blas_smat.c, 155
fasp_blas_smat_ymAx_nc3
blas_smat.c, 156
fasp_blas_smat_ymAx_nc5
blas_smat.c, 156
fasp_blas_smat_ymAx_nc7
blas_smat.c, 156
fasp_blas_smat_ymAx_ns
blas_smat.c, 157
fasp_blas_smat_ymAx_ns2
blas_smat.c, 157
fasp_blas_smat_ymAx_ns3
blas_smat.c, 158
fasp_blas_smat_ymAx_ns5
blas_smat.c, 158
fasp_blas_smat_ymAx_ns7
blas_smat.c, 159
fasp_blas_smat_ypAx
blas_smat.c, 159
fasp_blas_smat_ypAx_nc2
blas_smat.c, 160
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, 197
block_BSR, 199
block_Reservoir, 199
block_dCSRmat, 199
block_dvector, 199
block_iCSRmat, 199
block_ivector, 199
dBSRmat, 199
precond_block_reservoir_data, 199
fasp_check_dCSRmat
checkmat.c, 172
fasp_check_diagdom
checkmat.c, 173
fasp_check_diagpos
checkmat.c, 174
fasp_check_diagzero
checkmat.c, 175
fasp_check_iCSRmat
checkmat.c, 175
fasp_check_symm
checkmat.c, 175
fasp_chkerr
message.c, 361
fasp_const.h, 199
AMLI_CYCLE, 203
ASCEND, 203
BIGREAL, 203
CF_ORDER, 203
CGPT, 203
CLASSIC_AMG, 203
COARSE_AC, 204
COARSE_CR, 204
COARSE_MIS, 204
COARSE_RS, 204
CPFIRST, 204
DESCEND, 204
ERROR_ALLOC_MEM, 204
ERROR_AMG_COARSE_TYPE, 204
ERROR_AMG_COARSEING, 205
ERROR_AMG_INTERP_TYPE, 205
ERROR_AMG_SMOOTH_TYPE, 205
ERROR_DATA_STRUCTURE, 205
ERROR_DATA_ZERODIAG, 205
ERROR_DUMMY_VAR, 205
ERROR_INPUT_PAR, 205

ERROR_LIC_TYPE, 205
 ERROR_MAT_SIZE, 205
 ERROR_MISC, 206
 ERROR_NUM_BLOCKS, 206
 ERROR_OPEN_FILE, 206
 ERROR_QUAD_DIM, 206
 ERROR_QUAD_TYPE, 206
 ERROR_REGRESS, 206
 ERROR_SOLVER_EXIT, 206
 ERROR_SOLVER_ILUSETUP, 206
 ERROR_SOLVER_MAXIT, 206
 ERROR_SOLVER_MISC, 207
 ERROR_SOLVER_PRECTYPE, 207
 ERROR_SOLVER_SOLSTAG, 207
 ERROR_SOLVER_STAG, 207
 ERROR_SOLVER_TOLSMALL, 207
 ERROR_SOLVER_TYPE, 207
 ERROR_UNKNOWN, 207
 ERROR_WRONG_FILE, 207
 FALSE, 207
 FASP_SUCCESS, 208
 FGPT, 208
 PPFIRST, 208
 G0PT, 208
 ILUK, 208
 ILUt, 208
 ILUtp, 208
 INTERP_DIR, 208
 INTERP_ENG, 209
 INTERP_STD, 209
 ISPT, 209
 MAT_BSR, 209
 MAT_CSR, 209
 MAT_CSRL, 209
 MAT_FREE, 210
 MAT_STR, 210
 MAT_SymCSR, 210
 MAT_bBSR, 209
 MAT_bCSR, 209
 MAX_AMG_LVL, 210
 MAX_CREATE, 210
 MAX_REFINE_LVL, 210
 MAX_RESTART, 210
 MAX_STAG, 210
 MIN_CDOF, 211
 MIN_CREATE, 211
 NL_AMLI_CYCLE, 211
 NO_ORDER, 211
 OFF, 211
 ON, 211
 OPENMP HOLDS, 211
 PAIRWISE, 211
 PREC_AMG, 212
 PREC_DIAG, 212
 PREC_FMG, 212
 PREC_ILU, 212
 PREC_NULL, 212
 PREC_SCHWARZ, 212
 PRINT_ALL, 212
 PRINT_MIN, 212
 PRINT_MORE, 213
 PRINT_MOST, 213
 PRINT_NONE, 213
 PRINT_SOME, 213
 SA_AMG, 213
 SMALLREAL, 213
 SMOOTHER_BLKOIL, 213
 SMOOTHER(CG, 213
 SMOOTHER_GS, 214
 SMOOTHER_GSOR, 214
 SMOOTHER_JACOBI, 214
 SMOOTHER_L1DIAG, 214
 SMOOTHER_POLY, 214
 SMOOTHER_SGS, 214
 SMOOTHER_SGSOR, 214
 SMOOTHER_SOR, 214
 SMOOTHER_SPETEN, 215
 SMOOTHER_SSOR, 215
 SOLVER_AMG, 215
 SOLVER_BICGstab, 215
 SOLVER(CG, 215
 SOLVER_DEFAULT, 215
 SOLVER_FMG, 215
 SOLVER_GCG, 215
 SOLVER_GCR, 216
 SOLVER_GMRES, 216
 SOLVER_MUMPS, 216
 SOLVER_MinRes, 216
 SOLVER_SBICGstab, 216
 SOLVER_SCG, 216
 SOLVER_SGCG, 216
 SOLVER_SGMRES, 216
 SOLVER_SMinRes, 216
 SOLVER_SUPERLU, 217
 SOLVER_SVFGMRES, 217
 SOLVER_SVGMRES, 217
 SOLVER_UMFPACK, 217
 SOLVER_VFGMRES, 217
 SOLVER_VGMRES, 217
 STAG_RATIO, 217
 STOP_MOD_REL_RES, 217
 STOP_REL_PRECRES, 217
 STOP_REL_RES, 218
 TRUE, 218
 UA_AMG, 218
 UNPT, 218
 USERDEFINED, 218
 V_CYCLE, 218

VMB, 218
W_CYCLE, 218
fasp_dbsr_Linfinity_dcsr
 sparse_block.c, 558
fasp_dbsr_alloc
 sparse_bsr.c, 561
fasp_dbsr_cp
 sparse_bsr.c, 562
fasp_dbsr_create
 sparse_bsr.c, 562
fasp_dbsr_diagLU
 sparse_bsr.c, 567
fasp_dbsr_diagLU2
 sparse_bsr.c, 568
fasp_dbsr_diaginv
 sparse_bsr.c, 563
fasp_dbsr_diaginv2
 sparse_bsr.c, 564
fasp_dbsr_diaginv3
 sparse_bsr.c, 565
fasp_dbsr_diaginv4
 sparse_bsr.c, 566
fasp_dbsr_diagpref
 sparse_bsr.c, 569
fasp_dbsr_free
 sparse_bsr.c, 570
fasp_dbsr_getblk
 sparse_block.c, 556
fasp_dbsr_getblk_dcsr
 sparse_block.c, 557
fasp_dbsr_getdiag
 sparse_bsr.c, 571
fasp_dbsr_getdiaginv
 sparse_bsr.c, 571
fasp_dbsr_null
 sparse_bsr.c, 572
fasp_dbsr_plot
 graphics.c, 241
fasp_dbsr_print
 io.c, 278
fasp_dbsr_read
 io.c, 278
fasp_dbsr_subplot
 graphics.c, 242
fasp_dbsr_trans
 sparse_bsr.c, 572
fasp_dbsr_write
 io.c, 279
fasp_dbsr_write_coo
 io.c, 280
fasp_dcoo1_read
 io.c, 280
fasp_dcoo_alloc
 sparse_coo.c, 574
fasp_dcoo_create
 sparse_coo.c, 574
fasp_dcoo_free
 sparse_coo.c, 575
fasp_dcoo_print
 io.c, 282
fasp_dcoo_read
 io.c, 282
fasp_dcoo_shift
 sparse_coo.c, 576
fasp_dcoo_shift_read
 io.c, 284
fasp_dcoo_write
 io.c, 285
fasp_dcsr_CMK_order
 ordering.c, 373
fasp_dcsr_RCMK_order
 ordering.c, 374
fasp_dcsr_Schwarz_backward_smoothen
 schwarz_setup.c, 493
fasp_dcsr_Schwarz_forward_smoothen
 schwarz_setup.c, 494
fasp_dcsr_alloc
 sparse_csr.c, 578
fasp_dcsr_compress
 sparse_csr.c, 579
fasp_dcsr_compress_inplace
 sparse_csr.c, 580
fasp_dcsr_cp
 sparse_csr.c, 581
fasp_dcsr_create
 sparse_csr.c, 581
fasp_dcsr_diagpref
 sparse_csr.c, 583
fasp_dcsr_eig
 eigen.c, 185
fasp_dcsr_free
 sparse_csr.c, 584
fasp_dcsr_getblk
 sparse_block.c, 559
fasp_dcsr_getcol
 sparse_csr.c, 584
fasp_dcsr_getdiag
 sparse_csr.c, 585
fasp_dcsr_multicoloring
 sparse_csr.c, 586
fasp_dcsr_null
 sparse_csr.c, 586
fasp_dcsr_perm
 sparse_csr.c, 586
fasp_dcsr_plot
 graphics.c, 243
fasp_dcsr_print
 io.c, 286

fasp_dcsr_read
 io.c, 286
 fasp_dcsr_rediag
 sparse_csr.c, 587
 fasp_dcsr_shift
 sparse_csr.c, 588
 fasp_dcsr_sort
 sparse_csr.c, 588
 fasp_dcsr_subplot
 graphics.c, 244
 fasp_dcsr_symdiagscale
 sparse_csr.c, 589
 fasp_dcsr_sympat
 sparse_csr.c, 590
 fasp_dcsr_trans
 sparse_csr.c, 591
 fasp_dcsr_write_coo
 io.c, 287
 fasp_dcsrl_create
 sparse_csrl.c, 596
 fasp_dcsrl_free
 sparse_csrl.c, 597
 fasp_dCSRvec1_read
 io.c, 287
 fasp_dCSRvec1_write
 io.c, 288
 fasp_dCSRvec2_read
 io.c, 289
 fasp_dCSRvec2_write
 io.c, 290
 fasp_dmtx_read
 io.c, 291
 fasp_dmtxsym_read
 io.c, 292
 fasp_dstr_alloc
 sparse_str.c, 598
 fasp_dstr_cp
 sparse_str.c, 599
 fasp_dstr_create
 sparse_str.c, 599
 fasp_dstr_diagscale
 blas_str.c, 164
 fasp_dstr_free
 sparse_str.c, 601
 fasp_dstr_null
 sparse_str.c, 602
 fasp_dstr_print
 io.c, 293
 fasp_dstr_read
 io.c, 293
 fasp_dstr_write
 io.c, 294
 fasp_dvec_alloc
 vec.c, 650
 fasp_dvec_cp
 vec.c, 651
 fasp_dvec_create
 vec.c, 651
 fasp_dvec_free
 vec.c, 652
 fasp_dvec_isnan
 vec.c, 652
 fasp_dvec_maxdiff
 vec.c, 653
 fasp_dvec_null
 vec.c, 653
 fasp_dvec_print
 io.c, 295
 fasp_dvec_rand
 vec.c, 654
 fasp_dvec_read
 io.c, 295
 fasp_dvec_set
 vec.c, 654
 fasp_dvec_symdiagscale
 vec.c, 655
 fasp_dvec_write
 io.c, 296
 fasp_dvecind_read
 io.c, 297
 fasp_dvecind_write
 io.c, 298
 fasp_famg_solve
 amg_solve.c, 93
 fasp_format_bdcsr_dcsr
 formats.c, 222
 fasp_format_dbsr_dc oo
 formats.c, 222
 fasp_format_dbsr_dcsr
 formats.c, 223
 fasp_format_dc oo_dcsr
 formats.c, 223
 fasp_format_dCSR_dbsr
 formats.c, 225
 fasp_format_dCSR_dc oo
 formats.c, 226
 fasp_format_dcsrl_dcsr
 formats.c, 227
 fasp_format_dstr_dbsr
 formats.c, 227
 fasp_format_dstr_dcsr
 formats.c, 228
 fasp_fwrapper_amg_ wrapper.c, 660
 fasp_fwrapper_krylov_amg_ wrapper.c, 661
 fasp_gauss2d
 quadrature.c, 488

fasp_generate_diaginv_block
smoother_str.c, 539

fasp_gettime
timing.c, 648

fasp_grid2d_plot
graphics.c, 245

fasp_hb_read
io.c, 299

fasp_iarray_cp
array.c, 104

fasp_iarray_set
array.c, 104

fasp_icsr_cp
sparse_csr.c, 591

fasp_icsr_create
sparse_csr.c, 593

fasp_icsr_free
sparse_csr.c, 594

fasp_icsr_null
sparse_csr.c, 594

fasp_icsr_trans
sparse_csr.c, 595

fasp_iden_free
smat.c, 502

fasp_ilu_data_alloc
init.c, 258

fasp_ilu_data_free
init.c, 258

fasp_ilu_data_null
init.c, 259

fasp_ilu_dbsr_setup
ilu_setup_bsr.c, 247

fasp_ilu_dcsr_setup
ilu_setup_csr.c, 248

fasp_ilu_dstr_setup0
ilu_setup_str.c, 250

fasp_ilu_dstr_setup1
ilu_setup_str.c, 251

fasp_ivec_alloc
vec.c, 656

fasp_ivec_create
vec.c, 656

fasp_ivec_free
vec.c, 657

fasp_ivec_print
io.c, 300

fasp_ivec_read
io.c, 300

fasp_ivec_set
vec.c, 658

fasp_ivec_write
io.c, 301

fasp_ivecind_read
io.c, 302

fasp_matrix_read
io.c, 303

fasp_matrix_read_bin
io.c, 304

fasp_matrix_write
io.c, 305

fasp_mem_calloc
memory.c, 357

fasp_mem_check
memory.c, 357

fasp_mem_dCSR_check
memory.c, 357

fasp_mem_free
memory.c, 358

fasp_mem_iluData_check
memory.c, 358

fasp_mem_realloc
memory.c, 359

fasp_mem_usage
memory.c, 359

fasp_param_Schwarz_init
parameters.c, 381

fasp_param_Schwarz_print
parameters.c, 382

fasp_param_Schwarz_set
parameters.c, 382

fasp_param_amg_init
parameters.c, 376

fasp_param_amg_print
parameters.c, 376

fasp_param_amg_set
parameters.c, 377

fasp_param_amg_to_prec
parameters.c, 377

fasp_param_amg_to_prec_bsr
parameters.c, 377

fasp_param_check
input.c, 262

fasp_param_ilu_init
parameters.c, 378

fasp_param_ilu_print
parameters.c, 378

fasp_param_ilu_set
parameters.c, 378

fasp_param_init
parameters.c, 379

fasp_param_input
input.c, 263

fasp_param_input_init
parameters.c, 380

fasp_param_prec_to_amg
parameters.c, 381

fasp_param_prec_to_amg_bsr
parameters.c, 381

fasp_param_set
 parameters.c, 382
 fasp_param_solver_init
 parameters.c, 383
 fasp_param_solver_print
 parameters.c, 383
 fasp_param_solver_set
 parameters.c, 384
 fasp_poisson_fmgm_1D
 gmg_poisson.c, 231
 fasp_poisson_fmgm_2D
 gmg_poisson.c, 232
 fasp_poisson_fmgm_3D
 gmg_poisson.c, 233
 fasp_poisson_gmg_1D
 gmg_poisson.c, 234
 fasp_poisson_gmg_2D
 gmg_poisson.c, 235
 fasp_poisson_gmg_3D
 gmg_poisson.c, 236
 fasp_poisson_pcg_gmg_1D
 gmg_poisson.c, 237
 fasp_poisson_pcg_gmg_2D
 gmg_poisson.c, 238
 fasp_poisson_pcg_gmg_3D
 gmg_poisson.c, 239
 fasp_precond_Schwarz
 precond_csr.c, 459
 fasp_precond_amg
 precond_csr.c, 452
 fasp_precond_amg_nk
 precond_csr.c, 452
 fasp_precond_amli
 precond_csr.c, 453
 fasp_precond_block_diag_3
 precond_bcsr.c, 433
 fasp_precond_block_diag_3_amg
 precond_bcsr.c, 434
 fasp_precond_block_diag_4
 precond_bcsr.c, 435
 fasp_precond_block_lower_3
 precond_bcsr.c, 436
 fasp_precond_block_lower_3_amg
 precond_bcsr.c, 437
 fasp_precond_block_lower_4
 precond_bcsr.c, 438
 fasp_precond_data_null
 init.c, 259
 fasp_precond_dbsr_amg
 precond_bsr.c, 441
 fasp_precond_dbsr_amg_nk
 precond_bsr.c, 442
 fasp_precond_dbsr_diag
 precond_bsr.c, 443
 fasp_precond_dbsr_diag_nc2
 precond_bsr.c, 444
 fasp_precond_dbsr_diag_nc3
 precond_bsr.c, 445
 fasp_precond_dbsr_diag_nc5
 precond_bsr.c, 446
 fasp_precond_dbsr_diag_nc7
 precond_bsr.c, 447
 fasp_precond_dbsr_ilu
 precond_bsr.c, 448
 fasp_precond_dbsr_nl_amli
 precond_bsr.c, 449
 fasp_precond_diag
 precond_csr.c, 454
 fasp_precond_dstr_blockgs
 precond_str.c, 463
 fasp_precond_dstr_diag
 precond_str.c, 463
 fasp_precond_dstr_ilu0
 precond_str.c, 464
 fasp_precond_dstr_ilu0_backward
 precond_str.c, 465
 fasp_precond_dstr_ilu0_forward
 precond_str.c, 466
 fasp_precond_dstr_ilu1
 precond_str.c, 467
 fasp_precond_dstr_ilu1_backward
 precond_str.c, 468
 fasp_precond_dstr_ilu1_forward
 precond_str.c, 469
 fasp_precond_famg
 precond_csr.c, 455
 fasp_precond_free
 precond_csr.c, 456
 fasp_precond_ilu
 precond_csr.c, 457
 fasp_precond_ilu_backward
 precond_csr.c, 458
 fasp_precond_ilu_forward
 precond_csr.c, 458
 fasp_precond_nl_amli
 precond_csr.c, 459
 fasp_precond_null
 init.c, 259
 fasp_precond_setup
 precond_csr.c, 460
 fasp_precond_sweeping
 precond_bcsr.c, 439
 fasp_quad2d
 quadrature.c, 489
 fasp_set_GS_threads
 threads.c, 647
 fasp_smat_identity
 smat.c, 502

fasp_smat_identity_nc2
smat.c, 503
fasp_smat_identity_nc3
smat.c, 503
fasp_smat_identity_nc5
smat.c, 503
fasp_smat_identity_nc7
smat.c, 504
fasp_smat_lu_decomp
lu.c, 354
fasp_smat_lu_solve
lu.c, 355
fasp_smoothen_dbsr_gs
smoother_bsr.c, 506
fasp_smoothen_dbsr_gs1
smoother_bsr.c, 507
fasp_smoothen_dbsr_gs_ascend
smoother_bsr.c, 508
fasp_smoothen_dbsr_gs_ascend1
smoother_bsr.c, 509
fasp_smoothen_dbsr_gs_descend
smoother_bsr.c, 510
fasp_smoothen_dbsr_gs_descend1
smoother_bsr.c, 511
fasp_smoothen_dbsr_gs_order1
smoother_bsr.c, 512
fasp_smoothen_dbsr_gs_order2
smoother_bsr.c, 513
fasp_smoothen_dbsr_ilu
smoother_bsr.c, 514
fasp_smoothen_dbsr_jacobi
smoother_bsr.c, 515
fasp_smoothen_dbsr_jacobi1
smoother_bsr.c, 516
fasp_smoothen_dbsr_jacobi_setup
smoother_bsr.c, 517
fasp_smoothen_dbsr_sor
smoother_bsr.c, 518
fasp_smoothen_dbsr_sor1
smoother_bsr.c, 519
fasp_smoothen_dbsr_sor_ascend
smoother_bsr.c, 520
fasp_smoothen_dbsr_sor_descend
smoother_bsr.c, 521
fasp_smoothen_dbsr_sor_order
smoother_bsr.c, 522
fasp_smoothen_dcsr_L1diag
smoother_csr.c, 530
fasp_smoothen_dcsr_gs
smoother_csr.c, 525
fasp_smoothen_dcsr_gs_cf
smoother_csr.c, 525
fasp_smoothen_dcsr_gs_rb3d
smoother_csr.c, 526
fasp_smoothen_dcsr_gscr
smoother_csr_cr.c, 535
fasp_smoothen_dcsr_ilu
smoother_csr.c, 527
fasp_smoothen_dcsr_jacobi
smoother_csr.c, 527
fasp_smoothen_dcsr_kaczmarz
smoother_csr.c, 529
fasp_smoothen_dcsr_poly
smoother_csr_poly.c, 536
fasp_smoothen_dcsr_poly_old
smoother_csr_poly.c, 537
fasp_smoothen_dcsr_sgs
smoother_csr.c, 531
fasp_smoothen_dcsr_sor
smoother_csr.c, 532
fasp_smoothen_dcsr_sor_cf
smoother_csr.c, 533
fasp_smoothen_dstr_gs
smoother_str.c, 540
fasp_smoothen_dstr_gs1
smoother_str.c, 541
fasp_smoothen_dstr_gs_ascend
smoother_str.c, 542
fasp_smoothen_dstr_gs_cf
smoother_str.c, 543
fasp_smoothen_dstr_gs_descend
smoother_str.c, 544
fasp_smoothen_dstr_gs_order
smoother_str.c, 545
fasp_smoothen_dstr_jacobi
smoother_str.c, 546
fasp_smoothen_dstr_jacobi1
smoother_str.c, 547
fasp_smoothen_dstr_schwarz
smoother_str.c, 548
fasp_smoothen_dstr_sor
smoother_str.c, 549
fasp_smoothen_dstr_sor1
smoother_str.c, 550
fasp_smoothen_dstr_sor_ascend
smoother_str.c, 551
fasp_smoothen_dstr_sor_cf
smoother_str.c, 552
fasp_smoothen_dstr_sor_descend
smoother_str.c, 553
fasp_smoothen_dstr_sor_order
smoother_str.c, 554
fasp_solver_amg
amg.c, 78
fasp_solver_amli
amlirecur.c, 96
fasp_solver_bdcsr_itsolver
itsolver_bcsr.c, 309

fasp_solver_bdcsr_krylov
 itsolver_bcsr.c, 310
 fasp_solver_bdcsr_krylov_block_3
 itsolver_bcsr.c, 312
 fasp_solver_bdcsr_krylov_block_4
 itsolver_bcsr.c, 314
 fasp_solver_bdcsr_krylov_sweeping
 itsolver_bcsr.c, 316
 fasp_solver_bdcsr_pbcgs
 pbcgs.c, 386
 fasp_solver_bdcsr_pcg
 pcg.c, 397
 fasp_solver_bdcsr_pgmres
 pgmres.c, 414
 fasp_solver_bdcsr_pminres
 pminres.c, 423
 fasp_solver_bdcsr_pvfgmres
 pvfgmres.c, 471
 fasp_solver_bdcsr_pvgmres
 pvgmres.c, 480
 fasp_solver_bdcsr_spbcgs
 spbcgs.c, 612
 fasp_solver_bdcsr_spcg
 spcg.c, 619
 fasp_solver_bdcsr_spgmres
 spgmres.c, 626
 fasp_solver_bdcsr_spminres
 spminres.c, 634
 fasp_solver_bdcsr_spvgmres
 spvgmres.c, 639
 fasp_solver_dbsr_itsolver
 itsolver_bsr.c, 318
 fasp_solver_dbsr_krylov
 itsolver_bsr.c, 319
 fasp_solver_dbsr_krylov_amg
 itsolver_bsr.c, 321
 fasp_solver_dbsr_krylov_amg_nk
 itsolver_bsr.c, 322
 fasp_solver_dbsr_krylov_diag
 itsolver_bsr.c, 323
 fasp_solver_dbsr_krylov_ilu
 itsolver_bsr.c, 324
 fasp_solver_dbsr_krylov_nk_amg
 itsolver_bsr.c, 326
 fasp_solver_dbsr_pbcgs
 pbcgs.c, 387
 fasp_solver_dbsr_pcg
 pcg.c, 398
 fasp_solver_dbsr_pgmres
 pgmres.c, 415
 fasp_solver_dbsr_pvfgmres
 pvfgmres.c, 473
 fasp_solver_dbsr_pvgmres
 pvgmres.c, 480
 fasp_solver_dbsr_spbcgs
 spbcgs.c, 613
 fasp_solver_dbsr_spgmres
 spgmres.c, 626
 fasp_solver_dbsr_spvgmres
 spvgmres.c, 641
 fasp_solver_dcsr_itsolver
 itsolver_csr.c, 329
 fasp_solver_dcsr_krylov
 itsolver_csr.c, 330
 fasp_solver_dcsr_krylov_Schwarz
 itsolver_csr.c, 340
 fasp_solver_dcsr_krylov_amg
 itsolver_csr.c, 331
 fasp_solver_dcsr_krylov_amg_nk
 itsolver_csr.c, 333
 fasp_solver_dcsr_krylov_diag
 itsolver_csr.c, 335
 fasp_solver_dcsr_krylov_ilu
 itsolver_csr.c, 336
 fasp_solver_dcsr_krylov_ilu_M
 itsolver_csr.c, 337
 fasp_solver_dcsr_pbcgs
 pbcgs.c, 388
 fasp_solver_dcsr_pcg
 pcg.c, 400
 fasp_solver_dcsr_pgcr
 pgcr.c, 410
 fasp_solver_dcsr_pgcr1
 pgcr.c, 411
 fasp_solver_dcsr_pgmres
 pgmres.c, 416
 fasp_solver_dcsr_pminres
 pminres.c, 424
 fasp_solver_dcsr_pvfgmres
 pvfgmres.c, 475
 fasp_solver_dcsr_pvgmres
 pvgmres.c, 482
 fasp_solver_dcsr_spbcgs
 spbcgs.c, 614
 fasp_solver_dcsr_spcg
 spcg.c, 620
 fasp_solver_dcsr_spgmres
 spgmres.c, 627
 fasp_solver_dcsr_spminres
 spminres.c, 635
 fasp_solver_dcsr_spvgmres
 spvgmres.c, 642
 fasp_solver_dstr_itsolver
 itsolver_str.c, 346
 fasp_solver_dstr_krylov
 itsolver_str.c, 347

fasp_solver_dstr_krylov_blockgs
 itsolver_str.c, 349
fasp_solver_dstr_krylov_diag
 itsolver_str.c, 350
fasp_solver_dstr_krylov_ilu
 itsolver_str.c, 351
fasp_solver_dstr_pbcgss
 pbcgss.c, 391
fasp_solver_dstr_pcgs
 pcg.c, 401
fasp_solver_dstr_pgmres
 pgmres.c, 418
fasp_solver_dstr_pminres
 pminres.c, 427
fasp_solver_dstr_pvgmres
 pvgmres.c, 483
fasp_solver_dstr_spbcgss
 spbcgss.c, 615
fasp_solver_dstr_spcgs
 spcg.c, 623
fasp_solver_dstr_spgmres
 spgmres.c, 629
fasp_solver_dstr_spminres
 spminres.c, 636
fasp_solver_dstr_spvgmres
 spvgmres.c, 643
fasp_solver_famg
 famg.c, 187
fasp_solver_fmgcycle
 fmgcycle.c, 219
fasp_solver_itsolver
 itsolver_mf.c, 342
fasp_solver_itsolver_init
 itsolver_mf.c, 343
fasp_solver_krylov
 itsolver_mf.c, 344
fasp_solver_mgcycle
 mgcycle.c, 364
fasp_solver_mgcycle_bsr
 mgcycle.c, 365
fasp_solver_mgrecur
 mgrecur.c, 367
fasp_solver_mumps
 interface_mumps.c, 264
fasp_solver_mumps_steps
 interface_mumps.c, 265
fasp_solver_nl_amli
 amlirecur.c, 97
fasp_solver_nl_amli_bsr
 amlirecur.c, 98
fasp_solver_pbcgss
 pbcgss_mf.c, 394
fasp_solver_pcgs
 pcg_mf.c, 404
fasp_solver_pgpcg
 pgpcg_mf.c, 408
fasp_solver_pgmres
 pgmres_mf.c, 420
fasp_solver_pminres
 pminres_mf.c, 430
fasp_solver_pvfgmres
 pvfgmres_mf.c, 477
fasp_solver_pvgmres
 pvgmres_mf.c, 486
fasp_solver_superlu
 interface_superlu.c, 268
fasp_solver_umfpack
 interface_umfpack.c, 270
fasp_sparse_MIS
 sparse_util.c, 606
fasp_sparse_aat_
 sparse_util.c, 604
fasp_sparse_abyb_
 sparse_util.c, 604
fasp_sparse_abybms_
 sparse_util.c, 605
fasp_sparse_aplbums_
 sparse_util.c, 605
fasp_sparse_aplusb_
 sparse_util.c, 606
fasp_sparse_iit_
 sparse_util.c, 606
fasp_sparse_rapcmp_
 sparse_util.c, 607
fasp_sparse_raprms_
 sparse_util.c, 608
fasp_sparse_wta_
 sparse_util.c, 608
fasp_sparse_wtams_
 sparse_util.c, 609
fasp_sparse_ytx_
 sparse_util.c, 609
fasp_sparse_ytxbig_
 sparse_util.c, 610
fasp_vector_read
 io.c, 306
fasp_vector_write
 io.c, 307
fasp_wrapper_dbsr_krylov_amg
 wrapper.c, 662
fasp_wrapper_dcoo_dbsr_krylov_amg
 wrapper.c, 663
fmgcycle.c, 219
 fasp_solver_fmgcycle, 219
formats.c, 220
 fasp_format_bdcsl_dcsr, 222
 fasp_format_dbsr_dcoo, 222
 fasp_format_dbsr_dcsr, 223

fasp_format_dcoo_dcsr, 223
 fasp_format_dcsr_dbsr, 225
 fasp_format_dcsr_dcoo, 226
 fasp_format_dcsrl_dcsr, 227
 fasp_format_dstr_dbsr, 227
 fasp_format_dstr_dcsr, 228

 G0PT
 fasp_const.h, 208

 GE
 fasp.h, 192

 GT
 fasp.h, 192

 givens.c, 229
 fasp_aux_givens, 230

 gmg_poisson.c, 230
 fasp_poisson_fgmg_1D, 231
 fasp_poisson_fgmg_2D, 232
 fasp_poisson_fgmg_3D, 233
 fasp_poisson_gmg_1D, 234
 fasp_poisson_gmg_2D, 235
 fasp_poisson_gmg_3D, 236
 fasp_poisson_pcg_gmg_1D, 237
 fasp_poisson_pcg_gmg_2D, 238
 fasp_poisson_pcg_gmg_3D, 239

 graphics.c, 240
 fasp_dbsr_plot, 241
 fasp_dbsr_subplot, 242
 fasp_dcsr_plot, 243
 fasp_dcsr_subplot, 244
 fasp_grid2d_plot, 245

 grid2d, 36
 e, 36
 edges, 36
 ediri, 37
 efather, 37
 fasp.h, 195
 p, 37
 pdiri, 37
 pfather, 37
 s, 37
 t, 37
 tfather, 37
 triangles, 37
 vertices, 38

 iCOOmat, 38
 fasp.h, 195

 iCSRmat, 39
 fasp.h, 195

 ILU_data, 40

 ILU_droptol
 input_param, 46

 ILU_lfil
 input_param, 46

 ILU_param, 40

 ILU_permtol
 input_param, 46

 ILU_relax
 input_param, 46

 ILU_type
 input_param, 46

 ILUK
 fasp_const.h, 208

 ILUt
 fasp_const.h, 208

 ILUtp
 fasp_const.h, 208

 IMAP
 fasp.h, 196

 INT
 fasp.h, 193

 INTERP_DIR
 fasp_const.h, 208

 INTERP_ENG
 fasp_const.h, 209

 INTERP_STD
 fasp_const.h, 209

 ISNAN
 fasp.h, 193

 ISPT
 fasp_const.h, 209

 idenmat, 39
 fasp.h, 195

 ilength
 io.c, 308

 ilu.f, 245

 ilu_setup_bsr.c, 246
 fasp_ilu_dbsr_setup, 247

 ilu_setup_csr.c, 247
 fasp_ilu_dcsr_setup, 248

 ilu_setup_str.c, 249
 fasp_ilu_dstr_setup0, 250
 fasp_ilu_dstr_setup1, 251

 inifile
 input_param, 46

 init.c, 253
 fasp_Schwarz_data_free, 261
 fasp_amg_data_bsr_create, 255
 fasp_amg_data_bsr_free, 255
 fasp_amg_data_create, 256
 fasp_amg_data_free, 257
 fasp_ilu_data_alloc, 258
 fasp_ilu_data_free, 258
 fasp_ilu_data_null, 259
 fasp_precond_data_null, 259
 fasp_precond_null, 259

 input.c, 261
 fasp_param_check, 262

fasp_param_input, 263
input_param, 41
 AMG_ILU_levels, 43
 AMG_Schwarz_levels, 45
 AMG_aggregation_type, 42
 AMG_aggressive_level, 42
 AMG_aggressive_path, 43
 AMG_amli_degree, 43
 AMG_coarse_dof, 43
 AMG_coarse_scaling, 43
 AMG_coarse_solver, 43
 AMG_coarsening_type, 43
 AMG_cycle_type, 43
 AMG_interpolation_type, 43
 AMG_levels, 44
 AMG_max_aggregation, 44
 AMG_max_row_sum, 44
 AMG_maxit, 44
 AMG_nl_amli_krylov_type, 44
 AMG_pair_number, 44
 AMG_polynomial_degree, 44
 AMG_postsMOOTH_iter, 44
 AMG_presMOOTH_iter, 44
 AMG_relaxation, 45
 AMG_smooth_filter, 45
 AMG_smooth_order, 45
 AMG_smoothen, 45
 AMG_strong_coupled, 45
 AMG_strong_threshold, 45
 AMG_tentative_smooth, 45
 AMG_tol, 45
 AMG_truncation_threshold, 46
 AMG_type, 46
 ILU_droptol, 46
 ILU_lfil, 46
 ILU_permtol, 46
 ILU_relax, 46
 ILU_type, 46
 iniFILE, 46
 itsolver_maxit, 46
 itsolver_tol, 47
 output_type, 47
 precond_type, 47
 print_level, 47
 problem_num, 47
 restart, 47
 Schwarz_blkSOLVER, 47
 Schwarz_maxlvl, 47
 Schwarz_mmSize, 47
 Schwarz_type, 48
 solver_type, 48
 stop_type, 48
 workdir, 48
interface_mumps.c, 263
fasp_solver_mumps, 264
fasp_solver_mumps_steps, 265
interface_samg.c, 265
 dCSRmat2SAMGInput, 266
 dvector2SAMGInput, 267
interface_superlu.c, 268
 fasp_solver_superlu, 268
interface_umfpack.c, 269
 fasp_solver_umfpack, 270
interpolation.c, 271
 fasp_amg_interp, 272
 fasp_amg_interp1, 272
 fasp_amg_interp_trunc, 273
interpolation_em.c, 274
 fasp_amg_interp_em, 275
io.c, 275
 dlength, 308
 fasp_dbsr_print, 278
 fasp_dbsr_read, 278
 fasp_dbsr_write, 279
 fasp_dbsr_write_coo, 280
 fasp_dcOO1_read, 280
 fasp_dcOO_print, 282
 fasp_dcOO_read, 282
 fasp_dcOO_shift_read, 284
 fasp_dcOO_write, 285
 fasp_dCSR_print, 286
 fasp_dCSR_read, 286
 fasp_dCSR_write_coo, 287
 fasp_dCSRvec1_read, 287
 fasp_dCSRvec1_write, 288
 fasp_dCSRvec2_read, 289
 fasp_dCSRvec2_write, 290
 fasp_dMTX_read, 291
 fasp_dMTXsym_read, 292
 fasp_dSTR_print, 293
 fasp_dSTR_read, 293
 fasp_dSTR_write, 294
 fasp_dVec_print, 295
 fasp_dVec_read, 295
 fasp_dVec_write, 296
 fasp_dVecInd_read, 297
 fasp_dVecInd_write, 298
 fasp_hB_read, 299
 fasp_iVec_print, 300
 fasp_iVec_read, 300
 fasp_iVec_write, 301
 fasp_iVecInd_read, 302
 fasp_matrix_read, 303
 fasp_matrix_read_bin, 304
 fasp_matrix_write, 305
 fasp_vector_read, 306
 fasp_vector_write, 307
 ilength, 308

itsolver_bcsr.c, 308
 fasp_solver_bdcsr_itsolver, 309
 fasp_solver_bdcsr_krylov, 310
 fasp_solver_bdcsr_krylov_block_3, 312
 fasp_solver_bdcsr_krylov_block_4, 314
 fasp_solver_bdcsr_krylov_sweeping, 316

itsolver_bsr.c, 317
 fasp_solver_dbsr_itsolver, 318
 fasp_solver_dbsr_krylov, 319
 fasp_solver_dbsr_krylov_amg, 321
 fasp_solver_dbsr_krylov_amg_nk, 322
 fasp_solver_dbsr_krylov_diag, 323
 fasp_solver_dbsr_krylov_ilu, 324
 fasp_solver_dbsr_krylov_nk_amg, 326

itsolver_csr.c, 328
 fasp_solver_dcsr_itsolver, 329
 fasp_solver_dcsr_krylov, 330
 fasp_solver_dcsr_krylov_Schwarz, 340
 fasp_solver_dcsr_krylov_amg, 331
 fasp_solver_dcsr_krylov_amg_nk, 333
 fasp_solver_dcsr_krylov_diag, 335
 fasp_solver_dcsr_krylov_ilu, 336
 fasp_solver_dcsr_krylov_ilu_M, 337

itsolver_maxit
 input_param, 46

itsolver_mf.c, 341
 fasp_solver_itsolver, 342
 fasp_solver_itsolver_init, 343
 fasp_solver_krylov, 344

itsolver_param, 48
 itsolver_type, 49
 maxit, 49
 precond_type, 49
 print_level, 49
 restart, 49
 stop_type, 49
 tol, 49

itsolver_str.c, 345
 fasp_solver_dstr_itsolver, 346
 fasp_solver_dstr_krylov, 347
 fasp_solver_dstr_krylov_blockgs, 349
 fasp_solver_dstr_krylov_diag, 350
 fasp_solver_dstr_krylov_ilu, 351

itsolver_tol
 input_param, 47

itsolver_type
 itsolver_param, 49

ivector, 50
 fasp.h, 195

JA
 dBSRmat, 31

LE
 fasp.h, 193

LONG
 fasp.h, 193

LONGLONG
 fasp.h, 193

LS
 fasp.h, 193

LU_diag
 precond_block_data, 54

Link, 50

LinkList
 fasp.h, 195

linked_list, 51

ListElement
 fasp.h, 195

local_A
 precond_sweeping_data, 73

local_LU
 precond_sweeping_data, 73

local_index
 precond_sweeping_data, 73

lu.c, 353
 fasp_smat_lu_decomp, 354
 fasp_smat_lu_solve, 355

MAT_BSR
 fasp_const.h, 209

MAT_CSR
 fasp_const.h, 209

MAT_CSRL
 fasp_const.h, 209

MAT_FREE
 fasp_const.h, 210

MAT_STR
 fasp_const.h, 210

MAT_SymCSR
 fasp_const.h, 210

MAT_bBSR
 fasp_const.h, 209

MAT_bCSR
 fasp_const.h, 209

MAX
 fasp.h, 193

MAX_AMG_LVL
 fasp_const.h, 210

MAX_CREATE
 fasp_const.h, 210

MAX_REFINE_LVL
 fasp_const.h, 210

MAX_RESTART
 fasp_const.h, 210

MAX_STAG
 fasp_const.h, 210

MAXIMAP
 fasp.h, 196

MIN
 fasp.h, 193
MIN_CDOF
 fasp_const.h, 211
MIN_CRATE
 fasp_const.h, 211
maxit
 itsolver_param, 49
 precond_FASP_blkoi_data, 69
memory.c, 356
 fasp_mem_calloc, 357
 fasp_mem_check, 357
 fasp_mem_dcsr_check, 357
 fasp_mem_free, 358
 fasp_mem_iludata_check, 358
 fasp_mem_realloc, 359
 fasp_mem_usage, 359
 total_alloc_count, 360
 total_alloc_mem, 360
message.c, 360
 fasp_chkerr, 361
 print_amgcomplexity, 361
 print_amgcomplexity_bsr, 361
 print_cputime, 362
 print_itinfo, 362
 print_message, 363
mgcycle.c, 363
 fasp_solver_mgcycle, 364
 fasp_solver_mgcycle_bsr, 365
mgl
 precond_block_data, 54
mgl_data
 precond_FASP_blkoi_data, 69
mgrecur.c, 366
 fasp_solver_mgrecur, 367
Mumps_data, 51
mxv_matfree, 52

NEDMALLOC
 fasp.h, 193
NL_AMLI_CYCLE
 fasp_const.h, 211
NO_ORDER
 fasp_const.h, 211
neigh
 precond_FASP_blkoi_data, 69
NumLayers
 precond_sweeping_data, 73
nx_rb
 fasp.h, 196
ny_rb
 fasp.h, 196
nz_rb
 fasp.h, 196

OFF
 fasp_const.h, 211
ON
 fasp_const.h, 211
OPENMP HOLDS
 fasp_const.h, 211
order
 precond_FASP_blkoi_data, 69
 precond_block_reservoir_data, 57
ordering.c, 368
 fasp_BinarySearch, 373
 fasp_aux_dQuickSort, 370
 fasp_aux_dQuickSortIndex, 370
 fasp_aux_iQuickSort, 370
 fasp_aux_iQuickSortIndex, 371
 fasp_aux_merge, 371
 fasp_aux_msort, 372
 fasp_aux_unique, 373
 fasp_dcsr_CMK_order, 373
 fasp_dcsr_RCMK_order, 374
output_type
 input_param, 47

p
 grid2d, 37
PAIRWISE
 fasp_const.h, 211
PP
 precond_FASP_blkoi_data, 70
 precond_block_reservoir_data, 57
PREC_AMG
 fasp_const.h, 212
PREC_DIAG
 fasp_const.h, 212
PREC_FMGM
 fasp_const.h, 212
PREC_ILU
 fasp_const.h, 212
PREC_NULL
 fasp_const.h, 212
PREC_SCHWARZ
 fasp_const.h, 212
PRINT_ALL
 fasp_const.h, 212
PRINT_MIN
 fasp_const.h, 212
PRINT_MORE
 fasp_const.h, 213
PRINT_MOST
 fasp_const.h, 213
PRINT_NONE
 fasp_const.h, 213
PRINT_SOME
 fasp_const.h, 213

PRT_INT
 fasp.h, 194

PRT_REAL
 fasp.h, 194

parameters.c, 374
 fasp_param_Schwarz_init, 381
 fasp_param_Schwarz_print, 382
 fasp_param_Schwarz_set, 382
 fasp_param_amg_init, 376
 fasp_param_amg_print, 376
 fasp_param_amg_set, 377
 fasp_param_amg_to_prec, 377
 fasp_param_amg_to_prec_bsr, 377
 fasp_param_ilu_init, 378
 fasp_param_ilu_print, 378
 fasp_param_ilu_set, 378
 fasp_param_init, 379
 fasp_param_input_init, 380
 fasp_param_prec_to_amg, 381
 fasp_param_prec_to_amg_bsr, 381
 fasp_param_set, 382
 fasp_param_solver_init, 383
 fasp_param_solver_print, 383
 fasp_param_solver_set, 384

pbcgs.c, 384
 fasp_solver_bdcsr_pbcgs, 386
 fasp_solver_dbsr_pbcgs, 387
 fasp_solver_dcsr_pbcgs, 388
 fasp_solver_dstr_pbcgs, 391

pbcgs_mf.c, 392
 fasp_solver_pbcgs, 394

pcg.c, 395
 fasp_solver_bdcsr_pcg, 397
 fasp_solver_dbsr_pcg, 398
 fasp_solver_dcsr_pcg, 400
 fasp_solver_dstr_pcg, 401

pcg_mf.c, 403
 fasp_solver_pcg, 404

pcgrid2d
 fasp.h, 195

pdiri
 grid2d, 37

perf_idx
 precond_FASP_blkoil_data, 69
 precond_block_reservoir_data, 57

perf_neigh
 precond_FASP_blkoil_data, 70

pfather
 grid2d, 37

pgcg.c, 406
 fasp_solver_dcsr_pgcg, 406

pgcg_mf.c, 408
 fasp_solver_pgcg, 408

pgcr.c, 410

fasp_solver_dCSR_pgcr, 410
 fasp_solver_dCSR_pgcr1, 411

pgmres.c, 413
 fasp_solver_bdcsr_pgmres, 414
 fasp_solver_dbsr_pgmres, 415
 fasp_solver_dCSR_pgmres, 416
 fasp_solver_dstr_pgmres, 418

pgmres_mf.c, 419
 fasp_solver_pgmres, 420

pgrid2d
 fasp.h, 196

pivot
 precond_FASP_blkoil_data, 70
 precond_block_reservoir_data, 57

pivot_S
 precond_FASP_blkoil_data, 70

pivotS
 precond_block_reservoir_data, 57

pminres.c, 421
 fasp_solver_bdcsr_pminres, 423
 fasp_solver_dCSR_pminres, 424
 fasp_solver_dstr_pminres, 427

pminres_mf.c, 428
 fasp_solver_pminres, 430

precond, 52

precond_FASP_blkoil_data, 66
 A, 69
 diaginv, 69
 diaginv_S, 69
 diaginv_noscale, 69
 maxit, 69
 mgl_data, 69
 neigh, 69
 order, 69
 PP, 70
 perf_idx, 69
 perf_neigh, 70
 pivot, 70
 pivot_S, 70
 r, 70
 RR, 70
 restart, 70
 SS, 71
 scaled, 70
 tol, 71
 w, 71
 WW, 71

precond_bCSR.c, 431
 fasp_precond_block_diag_3, 433
 fasp_precond_block_diag_3_amg, 434
 fasp_precond_block_diag_4, 435
 fasp_precond_block_lower_3, 436
 fasp_precond_block_lower_3_amg, 437
 fasp_precond_block_lower_4, 438

fasp_precond_sweeping, 439
precond_block_data, 53
A_diag, 54
Abcsr, 54
amgparam, 54
LU_diag, 54
mgl, 54
r, 54
precond_block_reservoir_data, 55
diag, 57
diaginv, 57
diaginvS, 57
fasp_block.h, 199
order, 57
PP, 57
perf_idx, 57
pivot, 57
pivotS, 57
r, 58
RR, 58
SS, 58
scaled, 58
w, 58
WW, 58
precond_bsr.c, 440
fasp_precond_dbsr_amg, 441
fasp_precond_dbsr_amg_nk, 442
fasp_precond_dbsr_diag, 443
fasp_precond_dbsr_diag_nc2, 444
fasp_precond_dbsr_diag_nc3, 445
fasp_precond_dbsr_diag_nc5, 446
fasp_precond_dbsr_diag_nc7, 447
fasp_precond_dbsr_ilu, 448
fasp_precond_dbsr_nl_amli, 449
precond_csr.c, 450
fasp_precond_Schwarz, 459
fasp_precond_amg, 452
fasp_precond_amg_nk, 452
fasp_precond_amli, 453
fasp_precond_diag, 454
fasp_precond_famg, 455
fasp_precond_free, 456
fasp_precond_ilu, 457
fasp_precond_ilu_backward, 458
fasp_precond_ilu_forward, 458
fasp_precond_nl_amli, 459
fasp_precond_setup, 460
precond_data, 58
precond_data_bsr, 60
precond_data_str, 62
precond_diagbsr, 64
precond_diagstr, 65
precond_str.c, 462
fasp_precond_dstr_blockgs, 463
fasp_precond_dstr_diag, 463
fasp_precond_dstr_ilu0, 464
fasp_precond_dstr_ilu0_backward, 465
fasp_precond_dstr_ilu0_forward, 466
fasp_precond_dstr_ilu1, 467
fasp_precond_dstr_ilu1_backward, 468
fasp_precond_dstr_ilu1_forward, 469
precond_sweeping_data, 71
A, 73
Ai, 73
local_A, 73
local_LU, 73
local_index, 73
NumLayers, 73
r, 73
w, 73
precond_type
 input_param, 47
 itsolver_param, 49
print_amgcomplexity
 message.c, 361
print_amgcomplexity_bsr
 message.c, 361
print_cputime
 message.c, 362
print_itinfo
 message.c, 362
print_level
 input_param, 47
 itsolver_param, 49
print_message
 message.c, 363
problem_num
 input_param, 47
pvfgmres.c, 470
 fasp_solver_bdcsr_pvfgmres, 471
 fasp_solver_dbsr_pvfgmres, 473
 fasp_solver_dCSR_pvfgmres, 475
pvfgmres_mf.c, 477
 fasp_solver_pvfgmres, 477
pvgmres.c, 479
 fasp_solver_bdcsr_pvgmres, 480
 fasp_solver_dbsr_pvgmres, 480
 fasp_solver_dCSR_pvgmres, 482
 fasp_solver_dstr_pvgmres, 483
pvgmres_mf.c, 485
 fasp_solver_pvgmres, 486
quadrature.c, 487
 fasp_gauss2d, 488
 fasp_quad2d, 489
r
 precond_FASP_blkoi_data, 70
 precond_block_data, 54

precond_block_reservoir_data, 58
 precond_sweeping_data, 73
REAL
 fasp.h, 194
RR
 precond_FASP_blkoil_data, 70
 precond_block_reservoir_data, 58
RS_C1
 fasp.h, 194
rap.c, 490
 fasp blas dcsr_rap2, 490
restart
 input_param, 47
 itsolver_param, 49
 precond_FASP_blkoil_data, 70
s
 grid2d, 37
SA_AMG
 fasp_const.h, 213
SHORT
 fasp.h, 194
SMALLREAL
 fasp_const.h, 213
SMOOTHER_BLKOIL
 fasp_const.h, 213
SMOOTHER(CG)
 fasp_const.h, 213
SMOOTHER(GS)
 fasp_const.h, 214
SMOOTHER(GSOR)
 fasp_const.h, 214
SMOOTHER(JACOBI)
 fasp_const.h, 214
SMOOTHER(L1DIAG)
 fasp_const.h, 214
SMOOTHER(POLY)
 fasp_const.h, 214
SMOOTHER(SGS)
 fasp_const.h, 214
SMOOTHER(SGSOR)
 fasp_const.h, 214
SMOOTHER(SOR)
 fasp_const.h, 214
SMOOTHER(SPETEN)
 fasp_const.h, 215
SMOOTHER(SSOR)
 fasp_const.h, 215
SOLVER_AMG
 fasp_const.h, 215
SOLVER_BiCGstab
 fasp_const.h, 215
SOLVER(CG)
 fasp_const.h, 215
 SOLVER_DEFAULT
 fasp_const.h, 215
SOLVER_FM
 fasp_const.h, 215
SOLVER_GCG
 fasp_const.h, 215
SOLVER_GCR
 fasp_const.h, 216
SOLVER_GMRES
 fasp_const.h, 216
SOLVER_MUMPS
 fasp_const.h, 216
SOLVER_MinRes
 fasp_const.h, 216
SOLVER_SBICGSTAB
 fasp_const.h, 216
SOLVER_SCG
 fasp_const.h, 216
SOLVER_SCG
 fasp_const.h, 216
SOLVER_SGMRES
 fasp_const.h, 216
SOLVER_SMinRes
 fasp_const.h, 216
SOLVER_SUPERLU
 fasp_const.h, 217
SOLVER_SVFGMRES
 fasp_const.h, 217
SOLVER_SVGMRES
 fasp_const.h, 217
SOLVER_UMFPACK
 fasp_const.h, 217
SOLVER_VFGMRES
 fasp_const.h, 217
SOLVER_VGMRES
 fasp_const.h, 217
SS
 precond_FASP_blkoil_data, 71
 precond_block_reservoir_data, 58
STAG_RATIO
 fasp_const.h, 217
STOP_MOD_REL_RES
 fasp_const.h, 217
STOP_REL_PRECRES
 fasp_const.h, 217
STOP_REL_RES
 fasp_const.h, 218
scaled
 precond_FASP_blkoil_data, 70
 precond_block_reservoir_data, 58
schwarz.f, 491
Schwarz_blk solver
 input_param, 47
Schwarz_data, 74

Schwarz_maxlvl
 input_param, 47

Schwarz_mmsize
 input_param, 47

Schwarz_param, 75

schwarz_setup.c, 492
 fasp_Schwarz_get_block_matrix, 495
 fasp_Schwarz_setup, 496
 fasp_dcsr_Schwarz_backward_smoothen, 493
 fasp_dcsr_Schwarz_forward_smoothen, 494

Schwarz_type
 input_param, 48

smat.c, 497
 fasp blas smat Linfinity, 501
 fasp blas smat inv, 499
 fasp blas smat inv nc2, 499
 fasp blas smat inv nc3, 500
 fasp blas smat inv nc4, 500
 fasp blas smat inv nc5, 500
 fasp blas smat inv nc7, 501
 fasp iden free, 502
 fasp smat identity, 502
 fasp smat identity nc2, 503
 fasp smat identity nc3, 503
 fasp smat identity nc5, 503
 fasp smat identity nc7, 504

smoother_bsr.c, 504
 fasp smoother dbsr gs, 506
 fasp smoother dbsr gs1, 507
 fasp smoother dbsr gs ascend, 508
 fasp smoother dbsr gs ascend1, 509
 fasp smoother dbsr gs descend, 510
 fasp smoother dbsr gs descend1, 511
 fasp smoother dbsr gs order1, 512
 fasp smoother dbsr gs order2, 513
 fasp smoother dbsr ilu, 514
 fasp smoother dbsr jacobi, 515
 fasp smoother dbsr jacobi1, 516
 fasp smoother dbsr jacobi setup, 517
 fasp smoother dbsr sor, 518
 fasp smoother dbsr sor1, 519
 fasp smoother dbsr sor ascend, 520
 fasp smoother dbsr sor descend, 521
 fasp smoother dbsr sor order, 522

smoother_csr.c, 523
 fasp smoother dcsr L1diag, 530
 fasp smoother dcsr gs, 525
 fasp smoother dcsr gs cf, 525
 fasp smoother dcsr gs rb3d, 526
 fasp smoother dcsr ilu, 527
 fasp smoother dcsr jacobi, 527
 fasp smoother dcsr kaczmarz, 529
 fasp smoother dcsr sgs, 531
 fasp smoother dcsr sor, 532

 fasp smoother dcsr sor cf, 533
 smoother csr cr.c, 534
 fasp smoother dcsr gscr, 535
 smoother csr poly.c, 536
 fasp smoother dcsr poly, 536
 fasp smoother dcsr poly old, 537
smoother_str.c, 538
 fasp generate diaginv block, 539
 fasp smoother dstr gs, 540
 fasp smoother dstr gs1, 541
 fasp smoother dstr gs ascend, 542
 fasp smoother dstr gs cf, 543
 fasp smoother dstr gs descend, 544
 fasp smoother dstr gs order, 545
 fasp smoother dstr jacobi, 546
 fasp smoother dstr jacobi1, 547
 fasp smoother dstr schwarz, 548
 fasp smoother dstr sor, 549
 fasp smoother dstr sor1, 550
 fasp smoother dstr sor ascend, 551
 fasp smoother dstr sor cf, 552
 fasp smoother dstr sor descend, 553
 fasp smoother dstr sor order, 554

solver_type
 input_param, 48

sparse_block.c, 555
 fasp bdcsr free, 556
 fasp dbsr Linfinity dcsr, 558
 fasp dbsr getblk, 556
 fasp dbsr getblk dcsr, 557
 fasp dcsr getblk, 559

sparse_bsr.c, 560
 fasp dbsr alloc, 561
 fasp dbsr cp, 562
 fasp dbsr create, 562
 fasp dbsr diagLU, 567
 fasp dbsr diagLU2, 568
 fasp dbsr diaginv, 563
 fasp dbsr diaginv2, 564
 fasp dbsr diaginv3, 565
 fasp dbsr diaginv4, 566
 fasp dbsr diagpref, 569
 fasp dbsr free, 570
 fasp dbsr getdiag, 571
 fasp dbsr getdiaginv, 571
 fasp dbsr null, 572
 fasp dbsr trans, 572

sparse_coo.c, 573
 fasp dcoo alloc, 574
 fasp dcoo create, 574
 fasp dcoo free, 575
 fasp dcoo shift, 576

sparse_csr.c, 576
 fasp dcsr alloc, 578

fasp_dcsr_compress, 579
 fasp_dcsr_compress_inplace, 580
 fasp_dcsr_cp, 581
 fasp_dcsr_create, 581
 fasp_dcsr_diagpref, 583
 fasp_dcsr_free, 584
 fasp_dcsr_getcol, 584
 fasp_dcsr_getdiag, 585
 fasp_dcsr_multicoloring, 586
 fasp_dcsr_null, 586
 fasp_dcsr_perm, 586
 fasp_dcsr_rediag, 587
 fasp_dcsr_shift, 588
 fasp_dcsr_sort, 588
 fasp_dcsr_symdiagscale, 589
 fasp_dcsr_sympat, 590
 fasp_dcsr_trans, 591
 fasp_icsr_cp, 591
 fasp_icsr_create, 593
 fasp_icsr_free, 594
 fasp_icsr_null, 594
 fasp_icsr_trans, 595
sparse_csr1.c, 595
 fasp_dcsr1_create, 596
 fasp_dcsr1_free, 597
sparse_str.c, 597
 fasp_dstr_alloc, 598
 fasp_dstr_cp, 599
 fasp_dstr_create, 599
 fasp_dstr_free, 601
 fasp_dstr_null, 602
sparse_util.c, 602
 fasp_sparse_MIS, 606
 fasp_sparse_aat_, 604
 fasp_sparse_abyb_, 604
 fasp_sparse_abybms_, 605
 fasp_sparse_aplbms_, 605
 fasp_sparse_aplusb_, 606
 fasp_sparse_iit_, 606
 fasp_sparse_rapcmp_, 607
 fasp_sparse_rapms_, 608
 fasp_sparse_wta_, 608
 fasp_sparse_wtams_, 609
 fasp_sparse_ytx_, 609
 fasp_sparse_ytxbig_, 610
spbcgs.c, 610
 fasp_solver_bdcsr_spbcgs, 612
 fasp_solver_dbsr_spbcgs, 613
 fasp_solver_dCSR_spbcgs, 614
 fasp_solver_dstr_spbcgs, 615
spcg.c, 618
 fasp_solver_bdcsr_spcg, 619
 fasp_solver_dCSR_spcg, 620
 fasp_solver_dstr_spcg, 623
 spgmres.c, 625
 fasp_solver_bdcsr_spgmres, 626
 fasp_solver_dbsr_spgmres, 626
 fasp_solver_dCSR_spgmres, 627
 fasp_solver_dstr_spgmres, 629
spminres.c, 632
 fasp_solver_bdcsr_spminres, 634
 fasp_solver_dCSR_spminres, 635
 fasp_solver_dstr_spminres, 636
spvgmres.c, 638
 fasp_solver_bdcsr_spvgmres, 639
 fasp_solver_dbsr_spvgmres, 641
 fasp_solver_dCSR_spvgmres, 642
 fasp_solver_dstr_spvgmres, 643
stop_type
 input_param, 48
 itsolver_param, 49
t
 grid2d, 37
THDs_AMG_GS
 threads.c, 647
THDs_CPR_gGS
 threads.c, 647
THDs_CPR_IGS
 threads.c, 647
TRUE
 fasp_const.h, 218
tfather
 grid2d, 37
threads.c, 646
 FASP_GET_START_END, 646
 fasp_set_GS_threads, 647
THDs_AMG_GS, 647
THDs_CPR_gGS, 647
THDs_CPR_IGS, 647
timing.c, 648
 fasp_gettime, 648
tol
 itsolver_param, 49
 precond_FASP_blkoil_data, 71
total_alloc_count
 fasp.h, 196
 memory.c, 360
total_alloc_mem
 fasp.h, 196
 memory.c, 360
triangles
 grid2d, 37
UA_AMG
 fasp_const.h, 218
UNPT
 fasp_const.h, 218
USERDEFINED

fasp_const.h, 218

V_CYCLE
 fasp_const.h, 218

VMB
 fasp_const.h, 218

val
 dBSRmat, 31

vec.c, 649
 fasp_dvec_alloc, 650
 fasp_dvec_cp, 651
 fasp_dvec_create, 651
 fasp_dvec_free, 652
 fasp_dvec_isnan, 652
 fasp_dvec_maxdiff, 653
 fasp_dvec_null, 653
 fasp_dvec_rand, 654
 fasp_dvec_set, 654
 fasp_dvec_symdiagscale, 655
 fasp_ivec_alloc, 656
 fasp_ivec_create, 656
 fasp_ivec_free, 657
 fasp_ivec_set, 658

vertices
 grid2d, 38

w
 precond_FASP_blkoil_data, 71
 precond_block_reservoir_data, 58
 precond_sweeping_data, 73

W_CYCLE
 fasp_const.h, 218

WW
 precond_FASP_blkoil_data, 71
 precond_block_reservoir_data, 58

workdir
 input_param, 48

wrapper.c, 659
 fasp_fwrapper_amg_, 660
 fasp_fwrapper_krylov_amg_, 661
 fasp_wrapper_dbsr_krylov_amg, 662
 fasp_wrapper_dcoo_dbsr_krylov_amg, 663