

Fast Auxiliary Space Preconditioning

1.4.6 March/19/2014

Generated by Doxygen 1.8.6

Wed Mar 19 2014 11:47:33

Contents

1	Introduction	1
2	How to obtain FASP	3
3	Building and Installation	5
4	Developers	7
5	Doxygen	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	21
8.3	AMG_param Struct Reference	21
8.3.1	Detailed Description	23
8.4	block_BSR Struct Reference	23
8.4.1	Detailed Description	24
8.5	block_dCSRmat Struct Reference	24
8.5.1	Detailed Description	24
8.6	block_dvector Struct Reference	25
8.6.1	Detailed Description	25
8.7	block_iCSRmat Struct Reference	25
8.7.1	Detailed Description	25

8.8	block_ivector Struct Reference	26
8.8.1	Detailed Description	26
8.9	block_Reservoir Struct Reference	26
8.9.1	Detailed Description	26
8.10	dBSRmat Struct Reference	27
8.10.1	Detailed Description	27
8.10.2	Field Documentation	27
8.10.2.1	JA	27
8.10.2.2	val	28
8.11	dCOOmat Struct Reference	28
8.11.1	Detailed Description	28
8.12	dCSRLmat Struct Reference	29
8.12.1	Detailed Description	29
8.13	dCSRmat Struct Reference	29
8.13.1	Detailed Description	30
8.14	ddenmat Struct Reference	30
8.14.1	Detailed Description	30
8.15	dSTRmat Struct Reference	31
8.15.1	Detailed Description	31
8.16	dvector Struct Reference	31
8.16.1	Detailed Description	32
8.17	grid2d Struct Reference	32
8.17.1	Detailed Description	32
8.17.2	Field Documentation	33
8.17.2.1	e	33
8.17.2.2	edges	33
8.17.2.3	ediri	33
8.17.2.4	efather	33
8.17.2.5	p	33
8.17.2.6	pdiri	33
8.17.2.7	pfather	33
8.17.2.8	s	33
8.17.2.9	t	34
8.17.2.10	tfather	34
8.17.2.11	triangles	34
8.17.2.12	vertices	34
8.18	iCOOmat Struct Reference	34

8.18.1 Detailed Description	35
8.19 iCSRmat Struct Reference	35
8.19.1 Detailed Description	35
8.20 idenmat Struct Reference	36
8.20.1 Detailed Description	36
8.21 ILU_data Struct Reference	36
8.21.1 Detailed Description	37
8.22 ILU_param Struct Reference	37
8.22.1 Detailed Description	37
8.23 input_param Struct Reference	37
8.23.1 Detailed Description	39
8.23.2 Field Documentation	39
8.23.2.1 AMG_aggressive_level	39
8.23.2.2 AMG_aggressive_path	39
8.23.2.3 AMG_amli_degree	39
8.23.2.4 AMG_coarse_dof	39
8.23.2.5 AMG_coarse_scaling	39
8.23.2.6 AMG_coarsening_type	39
8.23.2.7 AMG_cycle_type	39
8.23.2.8 AMG_ILU_levels	40
8.23.2.9 AMG_interpolation_type	40
8.23.2.10 AMG_levels	40
8.23.2.11 AMG_max_aggregation	40
8.23.2.12 AMG_max_row_sum	40
8.23.2.13 AMG_maxit	40
8.23.2.14 AMG_nl_amli_krylov_type	40
8.23.2.15 AMG_polynomial_degree	40
8.23.2.16 AMG_postsmooth_iter	40
8.23.2.17 AMG_presmooth_iter	41
8.23.2.18 AMG_relaxation	41
8.23.2.19 AMG_schwarz_levels	41
8.23.2.20 AMG_smooth_filter	41
8.23.2.21 AMG_smooth_order	41
8.23.2.22 AMG_smoother	41
8.23.2.23 AMG_strong_coupled	41
8.23.2.24 AMG_strong_threshold	41
8.23.2.25 AMG_tentative_smooth	41

8.23.2.26	AMG_tol	42
8.23.2.27	AMG_truncation_threshold	42
8.23.2.28	AMG_type	42
8.23.2.29	ILU_droptol	42
8.23.2.30	ILU_lfil	42
8.23.2.31	ILU_permtol	42
8.23.2.32	ILU_relax	42
8.23.2.33	ILU_type	42
8.23.2.34	inifile	42
8.23.2.35	itsolver_maxit	43
8.23.2.36	itsolver_tol	43
8.23.2.37	output_type	43
8.23.2.38	precond_type	43
8.23.2.39	print_level	43
8.23.2.40	problem_num	43
8.23.2.41	restart	43
8.23.2.42	Schwarz_maxlvl	43
8.23.2.43	Schwarz_mmsize	43
8.23.2.44	Schwarz_type	44
8.23.2.45	solver_type	44
8.23.2.46	stop_type	44
8.23.2.47	workdir	44
8.24	itsolver_param Struct Reference	44
8.24.1	Detailed Description	44
8.24.2	Field Documentation	45
8.24.2.1	itsolver_type	45
8.24.2.2	maxit	45
8.24.2.3	precond_type	45
8.24.2.4	print_level	45
8.24.2.5	restart	45
8.24.2.6	stop_type	45
8.24.2.7	tol	45
8.25	ivector Struct Reference	45
8.25.1	Detailed Description	46
8.26	Link Struct Reference	46
8.26.1	Detailed Description	46
8.27	linked_list Struct Reference	46

8.27.1 Detailed Description	47
8.28 mxv_matfree Struct Reference	47
8.28.1 Detailed Description	47
8.29 precondition Struct Reference	48
8.29.1 Detailed Description	48
8.30 precondition_block_data Struct Reference	48
8.30.1 Detailed Description	48
8.30.2 Field Documentation	49
8.30.2.1 A	49
8.30.2.2 Aarray	49
8.30.2.3 Ablock	49
8.30.2.4 amgparam	49
8.30.2.5 col_idx	49
8.30.2.6 r	49
8.30.2.7 row_idx	49
8.31 precondition_block_reservoir_data Struct Reference	49
8.31.1 Detailed Description	51
8.31.2 Field Documentation	51
8.31.2.1 diag	51
8.31.2.2 diaginv	51
8.31.2.3 diaginvS	51
8.31.2.4 order	51
8.31.2.5 perf_idx	52
8.31.2.6 pivot	52
8.31.2.7 pivotS	52
8.31.2.8 PP	52
8.31.2.9 r	52
8.31.2.10 RR	52
8.31.2.11 scaled	52
8.31.2.12 SS	52
8.31.2.13 w	52
8.31.2.14 WW	53
8.32 precondition_data Struct Reference	53
8.32.1 Detailed Description	54
8.33 precondition_data_bsr Struct Reference	54
8.33.1 Detailed Description	55
8.34 precondition_data_str Struct Reference	56

8.34.1 Detailed Description	57
8.35 preconditioning Struct Reference	57
8.35.1 Detailed Description	57
8.36 preconditioning Struct Reference	58
8.36.1 Detailed Description	58
8.37 preconditioning_data Struct Reference	58
8.37.1 Detailed Description	60
8.37.2 Field Documentation	60
8.37.2.1 A	60
8.37.2.2 diaginv	60
8.37.2.3 diaginv_noscale	60
8.37.2.4 diaginv_S	60
8.37.2.5 maxit	60
8.37.2.6 mgl_data	60
8.37.2.7 neigh	61
8.37.2.8 order	61
8.37.2.9 perf_idx	61
8.37.2.10 perf_neigh	61
8.37.2.11 pivot	61
8.37.2.12 pivot_S	61
8.37.2.13 PP	61
8.37.2.14 r	61
8.37.2.15 restart	61
8.37.2.16 RR	62
8.37.2.17 scaled	62
8.37.2.18 SS	62
8.37.2.19 tol	62
8.37.2.20 w	62
8.37.2.21 WW	62
8.38 Schwarz_data Struct Reference	62
8.38.1 Detailed Description	63
8.39 Schwarz_param Struct Reference	63
8.39.1 Detailed Description	64
9 File Documentation	65
9.1 amg.c File Reference	65
9.1.1 Detailed Description	65

9.1.2	Function Documentation	65
9.1.2.1	<code>fasp_solver_amg</code>	65
9.2	<code>amg_setup_aggregation_bsr.inl</code> File Reference	66
9.2.1	Detailed Description	66
9.3	<code>amg_setup_aggregation_csr.inl</code> File Reference	66
9.3.1	Detailed Description	66
9.4	<code>amg_setup_cr.c</code> File Reference	66
9.4.1	Detailed Description	67
9.4.2	Function Documentation	67
9.4.2.1	<code>fasp_amg_setup_cr</code>	67
9.5	<code>amg_setup_rs.c</code> File Reference	67
9.5.1	Detailed Description	68
9.5.2	Function Documentation	68
9.5.2.1	<code>fasp_amg_setup_rs</code>	68
9.5.2.2	<code>fasp_amg_setup_rs_omp</code>	68
9.6	<code>amg_setup_sa.c</code> File Reference	69
9.6.1	Detailed Description	69
9.6.2	Function Documentation	69
9.6.2.1	<code>fasp_amg_setup_sa</code>	69
9.7	<code>amg_setup_ua.c</code> File Reference	70
9.7.1	Detailed Description	70
9.7.2	Function Documentation	71
9.7.2.1	<code>fasp_amg_setup_ua</code>	71
9.7.2.2	<code>fasp_amg_setup_ua_bsr</code>	71
9.8	<code>amg_solve.c</code> File Reference	71
9.8.1	Detailed Description	72
9.8.2	Function Documentation	72
9.8.2.1	<code>fasp_amg_solve</code>	72
9.8.2.2	<code>fasp_amg_solve_amli</code>	72
9.8.2.3	<code>fasp_amg_solve_nl_amli</code>	73
9.8.2.4	<code>fasp_famg_solve</code>	73
9.9	<code>amlirecur.c</code> File Reference	74
9.9.1	Detailed Description	74
9.9.2	Function Documentation	75
9.9.2.1	<code>fasp_amg_amli_coef</code>	75
9.9.2.2	<code>fasp_solver_amli</code>	76
9.9.2.3	<code>fasp_solver_nl_amli</code>	76

9.9.2.4	fasp_solver_nl_amli_bsr	77
9.10	array.c File Reference	77
9.10.1	Detailed Description	78
9.10.2	Function Documentation	78
9.10.2.1	fasp_array_cp	78
9.10.2.2	fasp_array_cp_nc3	78
9.10.2.3	fasp_array_cp_nc5	79
9.10.2.4	fasp_array_cp_nc7	79
9.10.2.5	fasp_array_null	79
9.10.2.6	fasp_array_set	80
9.10.2.7	fasp_iarray_cp	80
9.10.2.8	fasp_iarray_set	81
9.11	blas_array.c File Reference	81
9.11.1	Detailed Description	82
9.11.2	Function Documentation	82
9.11.2.1	fasp_blas_array_ax	82
9.11.2.2	fasp_blas_array_axpby	82
9.11.2.3	fasp_blas_array_axpy	83
9.11.2.4	fasp_blas_array_axpyz	83
9.11.2.5	fasp_blas_array_dotprod	84
9.11.2.6	fasp_blas_array_norm1	84
9.11.2.7	fasp_blas_array_norm2	85
9.11.2.8	fasp_blas_array_norminf	85
9.12	blas_bcsr.c File Reference	86
9.12.1	Detailed Description	86
9.12.2	Function Documentation	87
9.12.2.1	fasp_blas_bdbsr_aApy	87
9.12.2.2	fasp_blas_bdbsr_mxv	88
9.12.2.3	fasp_blas_bdcsl_aApy	88
9.12.2.4	fasp_blas_bdcsl_mxv	89
9.13	blas_bsr.c File Reference	89
9.13.1	Detailed Description	90
9.13.2	Function Documentation	90
9.13.2.1	fasp_blas_dbsr_aApy	90
9.13.2.2	fasp_blas_dbsr_aApy	90
9.13.2.3	fasp_blas_dbsr_aApy_agg	91
9.13.2.4	fasp_blas_dbsr_mxv	91

9.13.2.5	fasp_blas_dbsr_mxv_agg	92
9.13.2.6	fasp_blas_dbsr_rap	92
9.13.2.7	fasp_blas_dbsr_rap1	93
9.13.2.8	fasp_blas_dbsr_rap_agg	93
9.14	blas_csr.c File Reference	94
9.14.1	Detailed Description	95
9.14.2	Function Documentation	95
9.14.2.1	fasp_blas_dcsr_aApy	95
9.14.2.2	fasp_blas_dcsr_aApy_agg	96
9.14.2.3	fasp_blas_dcsr_add	97
9.14.2.4	fasp_blas_dcsr_axm	97
9.14.2.5	fasp_blas_dcsr_mxm	98
9.14.2.6	fasp_blas_dcsr_mxv	98
9.14.2.7	fasp_blas_dcsr_mxv_agg	99
9.14.2.8	fasp_blas_dcsr_ptap	99
9.14.2.9	fasp_blas_dcsr_rap	100
9.14.2.10	fasp_blas_dcsr_rap4	100
9.14.2.11	fasp_blas_dcsr_rap_agg	101
9.14.2.12	fasp_blas_dcsr_rap_agg1	101
9.14.2.13	fasp_blas_dcsr_vmv	102
9.15	blas_csrl.c File Reference	102
9.15.1	Detailed Description	103
9.15.2	Function Documentation	103
9.15.2.1	fasp_blas_dcsrl_mxv	103
9.16	blas_smat.c File Reference	103
9.16.1	Detailed Description	105
9.16.2	Function Documentation	105
9.16.2.1	fasp_blas_array_axpy_nc2	105
9.16.2.2	fasp_blas_array_axpy_nc3	105
9.16.2.3	fasp_blas_array_axpy_nc5	106
9.16.2.4	fasp_blas_array_axpy_nc7	106
9.16.2.5	fasp_blas_array_axpyz_nc2	106
9.16.2.6	fasp_blas_array_axpyz_nc3	107
9.16.2.7	fasp_blas_array_axpyz_nc5	107
9.16.2.8	fasp_blas_array_axpyz_nc7	108
9.16.2.9	fasp_blas_smat_aApyby	108
9.16.2.10	fasp_blas_smat_mul	109

9.16.2.11 fasp_blas_smat_mul_nc2	109
9.16.2.12 fasp_blas_smat_mul_nc3	110
9.16.2.13 fasp_blas_smat_mul_nc5	110
9.16.2.14 fasp_blas_smat_mul_nc7	110
9.16.2.15 fasp_blas_smat_mxv	111
9.16.2.16 fasp_blas_smat_mxv_nc2	111
9.16.2.17 fasp_blas_smat_mxv_nc3	112
9.16.2.18 fasp_blas_smat_mxv_nc5	113
9.16.2.19 fasp_blas_smat_mxv_nc7	113
9.16.2.20 fasp_blas_smat_ymAx	114
9.16.2.21 fasp_blas_smat_ymAx_nc2	115
9.16.2.22 fasp_blas_smat_ymAx_nc3	115
9.16.2.23 fasp_blas_smat_ymAx_nc5	116
9.16.2.24 fasp_blas_smat_ymAx_nc7	116
9.16.2.25 fasp_blas_smat_ymAx_ns	117
9.16.2.26 fasp_blas_smat_ymAx_ns2	117
9.16.2.27 fasp_blas_smat_ymAx_ns3	118
9.16.2.28 fasp_blas_smat_ymAx_ns5	118
9.16.2.29 fasp_blas_smat_ymAx_ns7	119
9.16.2.30 fasp_blas_smat_ypAx	119
9.16.2.31 fasp_blas_smat_ypAx_nc2	120
9.16.2.32 fasp_blas_smat_ypAx_nc3	120
9.16.2.33 fasp_blas_smat_ypAx_nc5	121
9.16.2.34 fasp_blas_smat_ypAx_nc7	122
9.17 blas_str.c File Reference	122
9.17.1 Detailed Description	123
9.17.2 Function Documentation	123
9.17.2.1 fasp_blas_dstr_aApy	123
9.17.2.2 fasp_blas_dstr_mxv	123
9.17.2.3 fasp_dstr_diagscale	123
9.18 blas_vec.c File Reference	124
9.18.1 Detailed Description	124
9.18.2 Function Documentation	125
9.18.2.1 fasp_blas_dvec_axpy	125
9.18.2.2 fasp_blas_dvec_axpyz	125
9.18.2.3 fasp_blas_dvec_dotprod	125
9.18.2.4 fasp_blas_dvec_norm1	126

9.18.2.5	fasp_blas_dvec_norm2	126
9.18.2.6	fasp_blas_dvec_norminf	127
9.18.2.7	fasp_blas_dvec_relerr	127
9.19	checkmat.c File Reference	128
9.19.1	Detailed Description	129
9.19.2	Function Documentation	129
9.19.2.1	fasp_check_dCSRmat	129
9.19.2.2	fasp_check_diagdom	129
9.19.2.3	fasp_check_diagpos	130
9.19.2.4	fasp_check_diagzero	131
9.19.2.5	fasp_check_iCSRmat	131
9.19.2.6	fasp_check_symm	132
9.20	coarsening_cr.c File Reference	133
9.20.1	Detailed Description	133
9.20.2	Function Documentation	133
9.20.2.1	fasp_amg_coarsening_cr	133
9.21	coarsening_rs.c File Reference	134
9.21.1	Detailed Description	134
9.21.2	Function Documentation	135
9.21.2.1	fasp_amg_coarsening_rs	135
9.22	convert.c File Reference	136
9.22.1	Detailed Description	137
9.22.2	Function Documentation	137
9.22.2.1	endian_convert_int	137
9.22.2.2	endian_convert_real	137
9.22.2.3	fasp_aux_bbyteToldouble	138
9.22.2.4	fasp_aux_change_endian4	139
9.22.2.5	fasp_aux_change_endian8	139
9.23	doxygen.h File Reference	140
9.23.1	Detailed Description	140
9.24	eigen.c File Reference	140
9.24.1	Detailed Description	140
9.24.2	Function Documentation	140
9.24.2.1	fasp_dcsr_eig	140
9.25	factor.f File Reference	141
9.25.1	Detailed Description	141
9.26	famg.c File Reference	141

9.26.1 Detailed Description	142
9.26.2 Function Documentation	142
9.26.2.1 fasp_solver_famg	142
9.27 fasp.h File Reference	142
9.27.1 Detailed Description	145
9.27.2 Macro Definition Documentation	145
9.27.2.1 __FASP_HEADER__	145
9.27.2.2 ABS	145
9.27.2.3 BIGREAL	145
9.27.2.4 C2N	146
9.27.2.5 DIAGONAL_PREF	146
9.27.2.6 DLMALLOC	146
9.27.2.7 FASP_GSRB	146
9.27.2.8 FASP_USE_ILU	146
9.27.2.9 GE	146
9.27.2.10 GT	146
9.27.2.11 INT	146
9.27.2.12 ISNAN	147
9.27.2.13 ISTART	147
9.27.2.14 LE	147
9.27.2.15 LONG	147
9.27.2.16 LONGLONG	147
9.27.2.17 LS	147
9.27.2.18 MAX	147
9.27.2.19 MAX_AMG_LVL	147
9.27.2.20 MAX_REFINE_LVL	148
9.27.2.21 MAX_RESTART	148
9.27.2.22 MAX_STAG	148
9.27.2.23 MIN	148
9.27.2.24 N2C	148
9.27.2.25 NEDMALLOC	148
9.27.2.26 OPENMP_HOLDS	148
9.27.2.27 REAL	148
9.27.2.28 RS_C1	148
9.27.2.29 SHORT	149
9.27.2.30 SMALLREAL	149
9.27.2.31 STAG_RATIO	149

9.27.3	Typedef Documentation	149
9.27.3.1	dCOOmat	149
9.27.3.2	dCSRLmat	149
9.27.3.3	dCSRmat	149
9.27.3.4	ddenmat	149
9.27.3.5	dSTRmat	149
9.27.3.6	dvector	149
9.27.3.7	grid2d	150
9.27.3.8	iCOOmat	150
9.27.3.9	iCSRmat	150
9.27.3.10	idenmat	150
9.27.3.11	ivector	150
9.27.3.12	LinkList	150
9.27.3.13	ListElement	150
9.27.3.14	pcgrid2d	150
9.27.3.15	pgrid2d	150
9.27.4	Variable Documentation	150
9.27.4.1	IMAP	150
9.27.4.2	MAXIMAP	151
9.27.4.3	nx_rb	151
9.27.4.4	ny_rb	151
9.27.4.5	nz_rb	151
9.27.4.6	total_alloc_count	151
9.27.4.7	total_alloc_mem	151
9.28	fasp_block.h File Reference	151
9.28.1	Detailed Description	152
9.28.2	Typedef Documentation	152
9.28.2.1	block_BSR	152
9.28.2.2	block_dCSRmat	153
9.28.2.3	block_dvector	153
9.28.2.4	block_iCSRmat	153
9.28.2.5	block_ivector	153
9.28.2.6	block_Reservoir	153
9.28.2.7	dBSRmat	153
9.28.2.8	precond_block_reservoir_data	153
9.29	fmgcycle.c File Reference	153
9.29.1	Detailed Description	154

9.29.2	Function Documentation	154
9.29.2.1	fasp_solver_fmecycle	154
9.30	formats.c File Reference	154
9.30.1	Detailed Description	155
9.30.2	Function Documentation	155
9.30.2.1	fasp_format_bdcsr_dcsr	155
9.30.2.2	fasp_format_dbsr_dcoo	155
9.30.2.3	fasp_format_dbsr_dcsr	156
9.30.2.4	fasp_format_dcoo_dcsr	156
9.30.2.5	fasp_format_dcsr_dbsr	157
9.30.2.6	fasp_format_dcsr_dcoo	157
9.30.2.7	fasp_format_dcsr_dcsr	157
9.30.2.8	fasp_format_dstr_dbsr	158
9.30.2.9	fasp_format_dstr_dcsr	158
9.31	givens.c File Reference	159
9.31.1	Detailed Description	159
9.31.2	Function Documentation	159
9.31.2.1	fasp_aux_givens	159
9.32	gmh_poisson.c File Reference	160
9.32.1	Detailed Description	160
9.32.2	Function Documentation	160
9.32.2.1	fasp_poisson_fgmg_1D	160
9.32.2.2	fasp_poisson_fgmg_2D	161
9.32.2.3	fasp_poisson_fgmg_3D	161
9.32.2.4	fasp_poisson_gmg_1D	162
9.32.2.5	fasp_poisson_gmg_2D	162
9.32.2.6	fasp_poisson_gmg_3D	163
9.32.2.7	fasp_poisson_pcg_gmg_1D	163
9.32.2.8	fasp_poisson_pcg_gmg_2D	164
9.32.2.9	fasp_poisson_pcg_gmg_3D	164
9.33	gmh_util.inl File Reference	165
9.33.1	Detailed Description	165
9.34	graphics.c File Reference	165
9.34.1	Detailed Description	166
9.34.2	Function Documentation	166
9.34.2.1	fasp_dbsr_plot	166
9.34.2.2	fasp_dcsr_subplot	166

9.34.2.3	fasp_grid2d_plot	167
9.35	ilu.f File Reference	167
9.35.1	Detailed Description	167
9.36	ilu_setup_bsr.c File Reference	168
9.36.1	Detailed Description	168
9.36.2	Function Documentation	168
9.36.2.1	fasp_ilu_dbsr_setup	168
9.37	ilu_setup_csr.c File Reference	169
9.37.1	Detailed Description	169
9.37.2	Function Documentation	169
9.37.2.1	fasp_ilu_dcsr_setup	169
9.38	ilu_setup_str.c File Reference	169
9.38.1	Detailed Description	170
9.38.2	Function Documentation	170
9.38.2.1	fasp_ilu_dstr_setup0	170
9.38.2.2	fasp_ilu_dstr_setup1	170
9.39	init.c File Reference	171
9.39.1	Detailed Description	172
9.39.2	Function Documentation	172
9.39.2.1	fasp_amg_data_bsr_create	172
9.39.2.2	fasp_amg_data_bsr_free	172
9.39.2.3	fasp_amg_data_create	172
9.39.2.4	fasp_amg_data_free	173
9.39.2.5	fasp_ilu_data_alloc	173
9.39.2.6	fasp_ilu_data_free	174
9.39.2.7	fasp_ilu_data_null	174
9.39.2.8	fasp_precond_data_null	174
9.39.2.9	fasp_precond_null	174
9.39.2.10	fasp_schwarz_data_free	175
9.40	input.c File Reference	175
9.40.1	Detailed Description	175
9.40.2	Function Documentation	176
9.40.2.1	fasp_param_check	176
9.40.2.2	fasp_param_input	176
9.41	interface_mumps.c File Reference	176
9.41.1	Detailed Description	177
9.41.2	Function Documentation	177

9.41.2.1	fasp_solver_mumps	177
9.41.2.2	fasp_solver_mumps_steps	177
9.42	interface_samg.c File Reference	178
9.42.1	Detailed Description	178
9.42.2	Function Documentation	178
9.42.2.1	dCSRmat2SAMGInput	178
9.42.2.2	dvector2SAMGInput	179
9.43	interface_superlu.c File Reference	180
9.43.1	Detailed Description	180
9.43.2	Function Documentation	180
9.43.2.1	fasp_solver_superlu	180
9.44	interface_umfpack.c File Reference	181
9.44.1	Detailed Description	181
9.44.2	Function Documentation	181
9.44.2.1	fasp_solver_umfpack	181
9.45	interpolation.c File Reference	182
9.45.1	Detailed Description	182
9.45.2	Function Documentation	182
9.45.2.1	fasp_amg_interp	182
9.45.2.2	fasp_amg_interp1	183
9.45.2.3	fasp_amg_interp_trunc	183
9.46	interpolation_em.c File Reference	184
9.46.1	Detailed Description	184
9.46.2	Function Documentation	184
9.46.2.1	fasp_amg_interp_em	184
9.47	io.c File Reference	185
9.47.1	Detailed Description	187
9.47.2	Function Documentation	187
9.47.2.1	fasp_dbsr_print	187
9.47.2.2	fasp_dbsr_read	187
9.47.2.3	fasp_dbsr_write	188
9.47.2.4	fasp_dbsr_write_coo	188
9.47.2.5	fasp_dcoo1_read	189
9.47.2.6	fasp_dcoo_print	189
9.47.2.7	fasp_dcoo_read	190
9.47.2.8	fasp_dcoo_write	190
9.47.2.9	fasp_dcsr_print	190

9.47.2.10	fasp_dcsr_read	191
9.47.2.11	fasp_dcsrvec1_read	191
9.47.2.12	fasp_dcsrvec1_write	192
9.47.2.13	fasp_dcsrvec2_read	192
9.47.2.14	fasp_dcsrvec2_write	193
9.47.2.15	fasp_dmtx_read	194
9.47.2.16	fasp_dmtxsym_read	194
9.47.2.17	fasp_dstr_print	195
9.47.2.18	fasp_dstr_read	195
9.47.2.19	fasp_dstr_write	196
9.47.2.20	fasp_dvec_print	196
9.47.2.21	fasp_dvec_read	197
9.47.2.22	fasp_dvec_write	197
9.47.2.23	fasp_dvecind_read	197
9.47.2.24	fasp_dvecind_write	198
9.47.2.25	fasp_ivec_print	198
9.47.2.26	fasp_ivec_read	199
9.47.2.27	fasp_ivec_write	199
9.47.2.28	fasp_ivecind_read	200
9.47.2.29	fasp_matrix_read	200
9.47.2.30	fasp_matrix_read_bin	201
9.47.2.31	fasp_matrix_write	202
9.47.2.32	fasp_vector_read	203
9.47.2.33	fasp_vector_write	204
9.48	itsolver_bcsr.c File Reference	205
9.48.1	Detailed Description	205
9.48.2	Function Documentation	205
9.48.2.1	fasp_solver_bdcscr_itsolver	205
9.48.2.2	fasp_solver_bdcscr_krylov	206
9.49	itsolver_bsr.c File Reference	206
9.49.1	Detailed Description	207
9.49.2	Function Documentation	207
9.49.2.1	fasp_set_GS_threads	207
9.49.2.2	fasp_solver_dbsr_itsolver	207
9.49.2.3	fasp_solver_dbsr_krylov	208
9.49.2.4	fasp_solver_dbsr_krylov_amg	208
9.49.2.5	fasp_solver_dbsr_krylov_diag	209

9.49.2.6	fasp_solver_dbsr_krylov_ilu	209
9.49.3	Variable Documentation	210
9.49.3.1	THDs_AMG_GS	210
9.49.3.2	THDs_CPR_gGS	210
9.49.3.3	THDs_CPR_IGS	210
9.50	itsolver_csr.c File Reference	210
9.50.1	Detailed Description	211
9.50.2	Function Documentation	211
9.50.2.1	fasp_solver_dcsr_itsolver	211
9.50.2.2	fasp_solver_dcsr_krylov	212
9.50.2.3	fasp_solver_dcsr_krylov_amg	212
9.50.2.4	fasp_solver_dcsr_krylov_diag	213
9.50.2.5	fasp_solver_dcsr_krylov_ilu	213
9.50.2.6	fasp_solver_dcsr_krylov_ilu_M	214
9.50.2.7	fasp_solver_dcsr_krylov_schwarz	214
9.51	itsolver_mf.c File Reference	215
9.51.1	Detailed Description	215
9.51.2	Function Documentation	216
9.51.2.1	fasp_solver_itsolver	216
9.51.2.2	fasp_solver_itsolver_init	217
9.51.2.3	fasp_solver_krylov	217
9.52	itsolver_str.c File Reference	218
9.52.1	Detailed Description	219
9.52.2	Function Documentation	219
9.52.2.1	fasp_solver_dstr_itsolver	219
9.52.2.2	fasp_solver_dstr_krylov	219
9.52.2.3	fasp_solver_dstr_krylov_blockgs	220
9.52.2.4	fasp_solver_dstr_krylov_diag	220
9.52.2.5	fasp_solver_dstr_krylov_ilu	221
9.53	itsolver_util.inl File Reference	221
9.53.1	Detailed Description	222
9.54	linklist.inl File Reference	222
9.54.1	Detailed Description	222
9.54.2	Macro Definition Documentation	222
9.54.2.1	LIST_HEAD	222
9.54.2.2	LIST_TAIL	222
9.55	lu.c File Reference	223

9.55.1 Detailed Description	223
9.55.2 Function Documentation	223
9.55.2.1 fasp_smat_lu_decomp	223
9.55.2.2 fasp_smat_lu_solve	224
9.56 memory.c File Reference	224
9.56.1 Detailed Description	225
9.56.2 Function Documentation	225
9.56.2.1 fasp_mem_check	225
9.56.2.2 fasp_mem_dcsr_check	225
9.56.2.3 fasp_mem_free	226
9.56.2.4 fasp_mem_iludata_check	226
9.56.2.5 fasp_mem_realloc	227
9.56.2.6 fasp_mem_usage	227
9.56.3 Variable Documentation	227
9.56.3.1 total_alloc_count	227
9.56.3.2 total_alloc_mem	228
9.57 message.c File Reference	228
9.57.1 Detailed Description	228
9.57.2 Function Documentation	228
9.57.2.1 fasp_chkerr	228
9.57.2.2 print_amgcomplexity	229
9.57.2.3 print_amgcomplexity_bsr	229
9.57.2.4 print_cputime	229
9.57.2.5 print_itinfo	230
9.57.2.6 print_message	230
9.58 messages.h File Reference	231
9.58.1 Detailed Description	234
9.58.2 Macro Definition Documentation	234
9.58.2.1 AMLI_CYCLE	234
9.58.2.2 ASCEND	234
9.58.2.3 CF_ORDER	234
9.58.2.4 CGPT	234
9.58.2.5 CLASSIC_AMG	234
9.58.2.6 COARSE_AC	234
9.58.2.7 COARSE_CR	235
9.58.2.8 COARSE_RS	235
9.58.2.9 CPFIRST	235

9.58.2.10 DESCEND	235
9.58.2.11 ERROR_ALLOC_MEM	235
9.58.2.12 ERROR_AMG_COARSE_TYPE	235
9.58.2.13 ERROR_AMG_COARSEING	235
9.58.2.14 ERROR_AMG_INTERP_TYPE	235
9.58.2.15 ERROR_AMG_SMOOTH_TYPE	236
9.58.2.16 ERROR_DATA_STRUCTURE	236
9.58.2.17 ERROR_DATA_ZERODIAG	236
9.58.2.18 ERROR_DUMMY_VAR	236
9.58.2.19 ERROR_INPUT_PAR	236
9.58.2.20 ERROR_LIC_TYPE	236
9.58.2.21 ERROR_MISC	236
9.58.2.22 ERROR_NUM_BLOCKS	236
9.58.2.23 ERROR_OPEN_FILE	236
9.58.2.24 ERROR_QUAD_DIM	237
9.58.2.25 ERROR_QUAD_TYPE	237
9.58.2.26 ERROR_REGRESS	237
9.58.2.27 ERROR_SOLVER_EXIT	237
9.58.2.28 ERROR_SOLVER_ILUSETUP	237
9.58.2.29 ERROR_SOLVER_MAXIT	237
9.58.2.30 ERROR_SOLVER_MISC	237
9.58.2.31 ERROR_SOLVER_PRECTYPE	237
9.58.2.32 ERROR_SOLVER_SOLSTAG	238
9.58.2.33 ERROR_SOLVER_STAG	238
9.58.2.34 ERROR_SOLVER_TOLSMALL	238
9.58.2.35 ERROR_SOLVER_TYPE	238
9.58.2.36 ERROR_WRONG_FILE	238
9.58.2.37 FALSE	238
9.58.2.38 FGPT	238
9.58.2.39 FPFIRST	238
9.58.2.40 ILUk	238
9.58.2.41 ILUt	239
9.58.2.42 ILUtp	239
9.58.2.43 INTERP_DIR	239
9.58.2.44 INTERP_ENG	239
9.58.2.45 INTERP_STD	239
9.58.2.46 ISPT	239

9.58.2.47 MAT_bBSR	239
9.58.2.48 MAT_bCSR	239
9.58.2.49 MAT_BSR	240
9.58.2.50 MAT_CSR	240
9.58.2.51 MAT_CSRL	240
9.58.2.52 MAT_FREE	240
9.58.2.53 MAT_STR	240
9.58.2.54 MAT_SymCSR	240
9.58.2.55 NL_AMLI_CYCLE	240
9.58.2.56 NO_ORDER	240
9.58.2.57 OFF	241
9.58.2.58 ON	241
9.58.2.59 PREC_AMG	241
9.58.2.60 PREC_DIAG	241
9.58.2.61 PREC_FMG	241
9.58.2.62 PREC_ILU	241
9.58.2.63 PREC_NULL	241
9.58.2.64 PREC_SCHWARZ	241
9.58.2.65 PRINT_ALL	242
9.58.2.66 PRINT_MIN	242
9.58.2.67 PRINT_MORE	242
9.58.2.68 PRINT_MOST	242
9.58.2.69 PRINT_NONE	242
9.58.2.70 PRINT_SOME	242
9.58.2.71 RUN_FAIL	242
9.58.2.72 SA_AMG	242
9.58.2.73 SMOOTHER_BLKOil	243
9.58.2.74 SMOOTHER_CG	243
9.58.2.75 SMOOTHER_GS	243
9.58.2.76 SMOOTHER_GSOR	243
9.58.2.77 SMOOTHER_JACOBI	243
9.58.2.78 SMOOTHER_L1DIAG	243
9.58.2.79 SMOOTHER_POLY	243
9.58.2.80 SMOOTHER_SGS	243
9.58.2.81 SMOOTHER_SGSOR	244
9.58.2.82 SMOOTHER_SOR	244
9.58.2.83 SMOOTHER_SPETEN	244

9.58.2.84 SMOOTHER_SSOR	244
9.58.2.85 SOLVER_AMG	244
9.58.2.86 SOLVER_BiCGstab	244
9.58.2.87 SOLVER_CG	244
9.58.2.88 SOLVER_FMG	244
9.58.2.89 SOLVER_GCG	245
9.58.2.90 SOLVER_GMRES	245
9.58.2.91 SOLVER_MinRes	245
9.58.2.92 SOLVER_MUMPS	245
9.58.2.93 SOLVER_SBiCGstab	245
9.58.2.94 SOLVER_SCG	245
9.58.2.95 SOLVER_SGCG	245
9.58.2.96 SOLVER_SGMRES	245
9.58.2.97 SOLVER_SMinRes	245
9.58.2.98 SOLVER_SUPERLU	246
9.58.2.99 SOLVER_SVFGMRES	246
9.58.2.100 SOLVER_SVGMRES	246
9.58.2.101 SOLVER_UMFPACK	246
9.58.2.102 SOLVER_VFGMRES	246
9.58.2.103 SOLVER_VGMRES	246
9.58.2.104 STOP_MOD_REL_RES	246
9.58.2.105 STOP_REL_PRECRES	246
9.58.2.106 STOP_REL_RES	247
9.58.2.107 SUCCESS	247
9.58.2.108 TRUE	247
9.58.2.109 UA_AMG	247
9.58.2.110 UNPT	247
9.58.2.111 USERDEFINED	247
9.58.2.112 V_CYCLE	247
9.58.2.113 W_CYCLE	247
9.59 mg_util.inl File Reference	248
9.59.1 Detailed Description	248
9.60 mgcycle.c File Reference	248
9.60.1 Detailed Description	248
9.60.2 Function Documentation	248
9.60.2.1 fasp_solver_mgcycle	248
9.60.2.2 fasp_solver_mgcycle_bsr	249

9.61	mgrecur.c File Reference	249
9.61.1	Detailed Description	250
9.61.2	Function Documentation	250
9.61.2.1	fasp_solver_mgrecur	250
9.62	ordering.c File Reference	250
9.62.1	Detailed Description	251
9.62.2	Function Documentation	251
9.62.2.1	fasp_aux_dQuickSort	251
9.62.2.2	fasp_aux_dQuickSortIndex	251
9.62.2.3	fasp_aux_iQuickSort	252
9.62.2.4	fasp_aux_iQuickSortIndex	252
9.62.2.5	fasp_aux_merge	253
9.62.2.6	fasp_aux_msort	253
9.62.2.7	fasp_aux_unique	254
9.62.2.8	fasp_BinarySearch	255
9.63	parameters.c File Reference	255
9.63.1	Detailed Description	256
9.63.2	Function Documentation	257
9.63.2.1	fasp_param_amg_init	257
9.63.2.2	fasp_param_amg_print	258
9.63.2.3	fasp_param_amg_set	258
9.63.2.4	fasp_param_amg_to_prec	258
9.63.2.5	fasp_param_amg_to_prec_bsr	259
9.63.2.6	fasp_param_ilu_init	259
9.63.2.7	fasp_param_ilu_print	259
9.63.2.8	fasp_param_ilu_set	260
9.63.2.9	fasp_param_init	260
9.63.2.10	fasp_param_input_init	261
9.63.2.11	fasp_param_prec_to_amg	261
9.63.2.12	fasp_param_prec_to_amg_bsr	261
9.63.2.13	fasp_param_schwarz_init	262
9.63.2.14	fasp_param_schwarz_print	263
9.63.2.15	fasp_param_schwarz_set	263
9.63.2.16	fasp_param_set	263
9.63.2.17	fasp_param_solver_init	264
9.63.2.18	fasp_param_solver_print	264
9.63.2.19	fasp_param_solver_set	264

9.64	pbcgs.c File Reference	265
9.64.1	Detailed Description	265
9.64.2	Function Documentation	266
9.64.2.1	fasp_solver_bdcsl_pbcgs	266
9.64.2.2	fasp_solver_dbsl_pbcgs	267
9.64.2.3	fasp_solver_dcsr_pbcgs	268
9.64.2.4	fasp_solver_dstr_pbcgs	268
9.65	pbcgs_mf.c File Reference	269
9.65.1	Detailed Description	269
9.65.2	Function Documentation	270
9.65.2.1	fasp_solver_pbcgs	270
9.66	pcg.c File Reference	271
9.66.1	Detailed Description	272
9.66.2	Function Documentation	273
9.66.2.1	fasp_solver_bdcsl_pcg	273
9.66.2.2	fasp_solver_dcsr_pcg	274
9.66.2.3	fasp_solver_dstr_pcg	275
9.67	pcg_mf.c File Reference	275
9.67.1	Detailed Description	275
9.67.2	Function Documentation	276
9.67.2.1	fasp_solver_pcg	276
9.68	pgcg.c File Reference	277
9.68.1	Detailed Description	277
9.68.2	Function Documentation	278
9.68.2.1	fasp_solver_dcsr_pgcg	278
9.69	pgcg_mf.c File Reference	278
9.69.1	Detailed Description	279
9.69.2	Function Documentation	279
9.69.2.1	fasp_solver_pgcg	279
9.70	pgmres.c File Reference	280
9.70.1	Detailed Description	280
9.70.2	Function Documentation	280
9.70.2.1	fasp_solver_bdcsl_pgmres	280
9.70.2.2	fasp_solver_dbsl_pgmres	281
9.70.2.3	fasp_solver_dcsr_pgmres	282
9.70.2.4	fasp_solver_dstr_pgmres	282
9.71	pgmres_mf.c File Reference	283

9.71.1 Detailed Description	283
9.71.2 Function Documentation	284
9.71.2.1 fasp_solver_pgmres	284
9.72 pminres.c File Reference	284
9.72.1 Detailed Description	285
9.72.2 Function Documentation	286
9.72.2.1 fasp_solver_bdcsr_pminres	286
9.72.2.2 fasp_solver_dcsr_pminres	286
9.72.2.3 fasp_solver_dstr_pminres	287
9.73 pminres_mf.c File Reference	288
9.73.1 Detailed Description	288
9.73.2 Function Documentation	289
9.73.2.1 fasp_solver_pminres	289
9.74 precondition_bsr.c File Reference	290
9.74.1 Detailed Description	290
9.74.2 Function Documentation	290
9.74.2.1 fasp_precond_dbsr_amg	290
9.74.2.2 fasp_precond_dbsr_diag	291
9.74.2.3 fasp_precond_dbsr_diag_nc2	291
9.74.2.4 fasp_precond_dbsr_diag_nc3	292
9.74.2.5 fasp_precond_dbsr_diag_nc5	293
9.74.2.6 fasp_precond_dbsr_diag_nc7	294
9.74.2.7 fasp_precond_dbsr_ilu	295
9.74.2.8 fasp_precond_dbsr_nl_amli	296
9.75 precondition_csr.c File Reference	296
9.75.1 Detailed Description	297
9.75.2 Function Documentation	297
9.75.2.1 fasp_precond_amg	297
9.75.2.2 fasp_precond_amli	298
9.75.2.3 fasp_precond_diag	298
9.75.2.4 fasp_precond_famg	298
9.75.2.5 fasp_precond_free	299
9.75.2.6 fasp_precond_ilu	299
9.75.2.7 fasp_precond_ilu_backward	299
9.75.2.8 fasp_precond_ilu_forward	300
9.75.2.9 fasp_precond_nl_amli	300
9.75.2.10 fasp_precond_schwarz	301

9.75.2.11 fasp_precond_setup	302
9.76 precondition_str.c File Reference	302
9.76.1 Detailed Description	303
9.76.2 Function Documentation	303
9.76.2.1 fasp_precond_dstr_blockgs	303
9.76.2.2 fasp_precond_dstr_diag	303
9.76.2.3 fasp_precond_dstr_ilu0	304
9.76.2.4 fasp_precond_dstr_ilu0_backward	304
9.76.2.5 fasp_precond_dstr_ilu0_forward	305
9.76.2.6 fasp_precond_dstr_ilu1	306
9.76.2.7 fasp_precond_dstr_ilu1_backward	306
9.76.2.8 fasp_precond_dstr_ilu1_forward	307
9.77 pvfgmres.c File Reference	308
9.77.1 Detailed Description	308
9.77.2 Function Documentation	309
9.77.2.1 fasp_solver_dbsr_pvfgmres	309
9.77.2.2 fasp_solver_dcsr_pvfgmres	310
9.78 pvfgmres_mf.c File Reference	311
9.78.1 Detailed Description	311
9.78.2 Function Documentation	311
9.78.2.1 fasp_solver_pvfgmres	311
9.79 pvgmres.c File Reference	312
9.79.1 Detailed Description	313
9.79.2 Function Documentation	313
9.79.2.1 fasp_solver_bdcsr_pvgmres	313
9.79.2.2 fasp_solver_dbsr_pvgmres	314
9.79.2.3 fasp_solver_dcsr_pvgmres	315
9.79.2.4 fasp_solver_dstr_pvgmres	316
9.80 pvgmres_mf.c File Reference	316
9.80.1 Detailed Description	317
9.80.2 Function Documentation	317
9.80.2.1 fasp_solver_pvgmres	317
9.81 quadrature.c File Reference	318
9.81.1 Detailed Description	318
9.81.2 Function Documentation	318
9.81.2.1 fasp_gauss2d	318
9.81.2.2 fasp_quad2d	318

9.82 rap.c File Reference	319
9.82.1 Detailed Description	319
9.82.2 Function Documentation	319
9.82.2.1 fasp_blas_dcsr_rap2	319
9.83 schwarz.f File Reference	320
9.83.1 Detailed Description	320
9.84 schwarz_setup.c File Reference	321
9.84.1 Detailed Description	321
9.84.2 Function Documentation	321
9.84.2.1 fasp_schwarz_setup	321
9.85 smat.c File Reference	322
9.85.1 Detailed Description	323
9.85.2 Function Documentation	323
9.85.2.1 fasp_blas_smat_inv	323
9.85.2.2 fasp_blas_smat_inv_nc2	323
9.85.2.3 fasp_blas_smat_inv_nc3	323
9.85.2.4 fasp_blas_smat_inv_nc4	324
9.85.2.5 fasp_blas_smat_inv_nc5	324
9.85.2.6 fasp_blas_smat_inv_nc7	324
9.85.2.7 fasp_iden_free	325
9.85.2.8 fasp_smat_identity	325
9.85.2.9 fasp_smat_identity_nc2	326
9.85.2.10 fasp_smat_identity_nc3	327
9.85.2.11 fasp_smat_identity_nc5	327
9.85.2.12 fasp_smat_identity_nc7	327
9.86 smoother_bsr.c File Reference	328
9.86.1 Detailed Description	329
9.86.2 Function Documentation	329
9.86.2.1 fasp_smoother_dbsr_gs	329
9.86.2.2 fasp_smoother_dbsr_gs1	329
9.86.2.3 fasp_smoother_dbsr_gs_ascend	330
9.86.2.4 fasp_smoother_dbsr_gs_ascend1	330
9.86.2.5 fasp_smoother_dbsr_gs_descend	331
9.86.2.6 fasp_smoother_dbsr_gs_descend1	331
9.86.2.7 fasp_smoother_dbsr_gs_order1	332
9.86.2.8 fasp_smoother_dbsr_gs_order2	332
9.86.2.9 fasp_smoother_dbsr_ilu	333

9.86.2.10	fasp_smoother_dbsr_jacobi	333
9.86.2.11	fasp_smoother_dbsr_jacobi1	334
9.86.2.12	fasp_smoother_dbsr_jacobi_setup	335
9.86.2.13	fasp_smoother_dbsr_sor	335
9.86.2.14	fasp_smoother_dbsr_sor1	336
9.86.2.15	fasp_smoother_dbsr_sor_ascend	336
9.86.2.16	fasp_smoother_dbsr_sor_descend	337
9.86.2.17	fasp_smoother_dbsr_sor_order	337
9.87	smoother_csr.c File Reference	338
9.87.1	Detailed Description	339
9.87.2	Function Documentation	339
9.87.2.1	fasp_smoother_dcsr_gs	339
9.87.2.2	fasp_smoother_dcsr_gs_cf	339
9.87.2.3	fasp_smoother_dcsr_ilu	340
9.87.2.4	fasp_smoother_dcsr_jacobi	340
9.87.2.5	fasp_smoother_dcsr_kaczmarz	341
9.87.2.6	fasp_smoother_dcsr_L1diag	341
9.87.2.7	fasp_smoother_dcsr_sgs	342
9.87.2.8	fasp_smoother_dcsr_sor	342
9.87.2.9	fasp_smoother_dcsr_sor_cf	343
9.88	smoother_csr_cr.c File Reference	343
9.88.1	Detailed Description	344
9.88.2	Function Documentation	344
9.88.2.1	fasp_smoother_dcsr_gscr	344
9.89	smoother_csr_poly.c File Reference	345
9.89.1	Detailed Description	345
9.89.2	Function Documentation	345
9.89.2.1	fasp_smoother_dcsr_poly	345
9.89.2.2	fasp_smoother_dcsr_poly_old	345
9.90	smoother_str.c File Reference	346
9.90.1	Detailed Description	347
9.90.2	Function Documentation	347
9.90.2.1	fasp_generate_diaginv_block	347
9.90.2.2	fasp_smoother_dstr_gs	347
9.90.2.3	fasp_smoother_dstr_gs1	348
9.90.2.4	fasp_smoother_dstr_gs_ascend	348
9.90.2.5	fasp_smoother_dstr_gs_cf	349

9.90.2.6	fasp_smoother_dstr_gs_descend	349
9.90.2.7	fasp_smoother_dstr_gs_order	350
9.90.2.8	fasp_smoother_dstr_jacobi	350
9.90.2.9	fasp_smoother_dstr_jacobi1	351
9.90.2.10	fasp_smoother_dstr_schwarz	351
9.90.2.11	fasp_smoother_dstr_sor	351
9.90.2.12	fasp_smoother_dstr_sor1	352
9.90.2.13	fasp_smoother_dstr_sor_ascend	352
9.90.2.14	fasp_smoother_dstr_sor_cf	353
9.90.2.15	fasp_smoother_dstr_sor_descend	353
9.90.2.16	fasp_smoother_dstr_sor_order	354
9.91	sparse_block.c File Reference	354
9.91.1	Detailed Description	355
9.91.2	Function Documentation	355
9.91.2.1	fasp_dbsr_getblk	355
9.91.2.2	fasp_dbsr_getblk_dcsr	355
9.91.2.3	fasp_dcsr_getblk	356
9.92	sparse_bsr.c File Reference	356
9.92.1	Detailed Description	357
9.92.2	Function Documentation	357
9.92.2.1	fasp_dbsr_alloc	357
9.92.2.2	fasp_dbsr_cp	358
9.92.2.3	fasp_dbsr_create	358
9.92.2.4	fasp_dbsr_diaginv	359
9.92.2.5	fasp_dbsr_diaginv2	359
9.92.2.6	fasp_dbsr_diaginv3	359
9.92.2.7	fasp_dbsr_diaginv4	360
9.92.2.8	fasp_dbsr_diagpref	360
9.92.2.9	fasp_dbsr_free	361
9.92.2.10	fasp_dbsr_getdiag	361
9.92.2.11	fasp_dbsr_getdiaginv	362
9.92.2.12	fasp_dbsr_null	362
9.92.2.13	fasp_dbsr_trans	363
9.93	sparse_coo.c File Reference	363
9.93.1	Detailed Description	364
9.93.2	Function Documentation	364
9.93.2.1	fasp_dcoo_alloc	364

9.93.2.2	fasp_dcoo_create	364
9.93.2.3	fasp_dcoo_free	364
9.93.2.4	fasp_dcoo_shift	365
9.94	sparse_csr.c File Reference	365
9.94.1	Detailed Description	366
9.94.2	Function Documentation	367
9.94.2.1	fasp_dcsr_alloc	367
9.94.2.2	fasp_dcsr_compress	367
9.94.2.3	fasp_dcsr_compress_inplace	367
9.94.2.4	fasp_dcsr_cp	368
9.94.2.5	fasp_dcsr_create	368
9.94.2.6	fasp_dcsr_diagpref	369
9.94.2.7	fasp_dcsr_free	369
9.94.2.8	fasp_dcsr_getcol	370
9.94.2.9	fasp_dcsr_getdiag	370
9.94.2.10	fasp_dcsr_multicoloring	371
9.94.2.11	fasp_dcsr_null	372
9.94.2.12	fasp_dcsr_perm	372
9.94.2.13	fasp_dcsr_regdiag	373
9.94.2.14	fasp_dcsr_shift	373
9.94.2.15	fasp_dcsr_sort	374
9.94.2.16	fasp_dcsr_symdiagscale	375
9.94.2.17	fasp_dcsr_sympat	375
9.94.2.18	fasp_dcsr_trans	376
9.94.2.19	fasp_icsr_cp	377
9.94.2.20	fasp_icsr_create	377
9.94.2.21	fasp_icsr_free	378
9.94.2.22	fasp_icsr_null	378
9.94.2.23	fasp_icsr_trans	378
9.95	sparse_csrl.c File Reference	379
9.95.1	Detailed Description	379
9.95.2	Function Documentation	379
9.95.2.1	fasp_dcsrl_create	379
9.95.2.2	fasp_dcsrl_free	380
9.96	sparse_str.c File Reference	380
9.96.1	Detailed Description	381
9.96.2	Function Documentation	381

9.96.2.1	fasp_dstr_alloc	381
9.96.2.2	fasp_dstr_cp	381
9.96.2.3	fasp_dstr_create	382
9.96.2.4	fasp_dstr_free	382
9.96.2.5	fasp_dstr_null	382
9.97	sparse_util.c File Reference	383
9.97.1	Detailed Description	384
9.97.2	Function Documentation	384
9.97.2.1	fasp_sparse_aat_	384
9.97.2.2	fasp_sparse_abyb_	384
9.97.2.3	fasp_sparse_abybms_	385
9.97.2.4	fasp_sparse_aplbms_	385
9.97.2.5	fasp_sparse_aplusb_	386
9.97.2.6	fasp_sparse_iit_	386
9.97.2.7	fasp_sparse_MIS	386
9.97.2.8	fasp_sparse_rapcmp_	387
9.97.2.9	fasp_sparse_rapms_	387
9.97.2.10	fasp_sparse_wta_	388
9.97.2.11	fasp_sparse_wtams_	388
9.97.2.12	fasp_sparse_ytx_	390
9.97.2.13	fasp_sparse_ytxbig_	390
9.98	spbcgs.c File Reference	391
9.98.1	Detailed Description	391
9.98.2	Function Documentation	392
9.98.2.1	fasp_solver_bdcsr_spbcgs	392
9.98.2.2	fasp_solver_dbsr_spbcgs	393
9.98.2.3	fasp_solver_dcsr_spbcgs	394
9.98.2.4	fasp_solver_dstr_spbcgs	395
9.99	spcg.c File Reference	396
9.99.1	Detailed Description	396
9.99.2	Function Documentation	397
9.99.2.1	fasp_solver_bdcsr_spcg	397
9.99.2.2	fasp_solver_dcsr_spcg	398
9.99.2.3	fasp_solver_dstr_spcg	398
9.100	spgmres.c File Reference	399
9.100.1	Detailed Description	400
9.100.2	Function Documentation	400

9.100.2.1 fasp_solver_bdcsl_spgmres	400
9.100.2.2 fasp_solver_dbsl_spgmres	400
9.100.2.3 fasp_solver_dcsr_spgmres	401
9.100.2.4 fasp_solver_dstr_spgmres	402
9.101 spminres.c File Reference	402
9.101.1 Detailed Description	403
9.101.2 Function Documentation	404
9.101.2.1 fasp_solver_bdcsl_spminres	404
9.101.2.2 fasp_solver_dcsr_spminres	404
9.101.2.3 fasp_solver_dstr_spminres	405
9.102 spvgmres.c File Reference	406
9.102.1 Detailed Description	406
9.102.2 Function Documentation	406
9.102.2.1 fasp_solver_bdcsl_spvgmres	406
9.102.2.2 fasp_solver_dbsl_spvgmres	407
9.102.2.3 fasp_solver_dcsr_spvgmres	408
9.102.2.4 fasp_solver_dstr_spvgmres	408
9.103 threads.c File Reference	409
9.103.1 Detailed Description	409
9.103.2 Function Documentation	409
9.103.2.1 FASP_GET_START_END	409
9.104 timing.c File Reference	410
9.104.1 Detailed Description	410
9.104.2 Function Documentation	410
9.104.2.1 fasp_gettime	410
9.105 vec.c File Reference	411
9.105.1 Detailed Description	411
9.105.2 Function Documentation	412
9.105.2.1 fasp_dvec_alloc	412
9.105.2.2 fasp_dvec_cp	412
9.105.2.3 fasp_dvec_create	412
9.105.2.4 fasp_dvec_free	413
9.105.2.5 fasp_dvec_isnan	413
9.105.2.6 fasp_dvec_maxdiff	413
9.105.2.7 fasp_dvec_null	414
9.105.2.8 fasp_dvec_rand	414
9.105.2.9 fasp_dvec_set	415

9.105.2.10	fasp_dvec_symdiagscale	415
9.105.2.11	fasp_ivec_alloc	416
9.105.2.12	fasp_ivec_create	416
9.105.2.13	fasp_ivec_free	417
9.105.2.14	fasp_ivec_set	418
9.106	wrapper.c File Reference	418
9.106.1	Detailed Description	419
9.106.2	Function Documentation	419
9.106.2.1	fasp_fwrapper_amg_	419
9.106.2.2	fasp_fwrapper_krylov_amg_	420
9.106.2.3	fasp_wrapper_dbsr_krylov_amg	420
9.106.2.4	fasp_wrapper_dcoo_dbsr_krylov_amg	421
	Index		422

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

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

will give you the developer version of the FASP package.

Chapter 3

Building and Installation

This is a simple instruction on building and testing. For more details, please refer to the README files and the short [User's Guide](#) in "faspolver/doc/".

To compile, you need a Fortran and a C compiler. First, you can type in the "faspolver/" 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 "faspolver/".

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)

Current Developers (in alphabetic order):

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

Doxygen

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

<http://www.doxygen.org>

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

Chapter 6

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	21
block_BSR	Block REAL matrix format for reservoir simulation	23
block_dCSRmat	Block REAL CSR matrix format	24
block_dvector	Block REAL vector structure	25
block_iCSRmat	Block INT CSR matrix format	25
block_ivector	Block INT vector structure	26
block_Reservoir	Block REAL matrix format for reservoir simulation	26
dBSRmat	Block sparse row storage matrix of REAL type	27
dCOOmat	Sparse matrix of REAL type in COO (or IJ) format	28
dCSRLmat	Sparse matrix of REAL type in CSRL format	29
dCSRmat	Sparse matrix of REAL type in CSR format	29
ddenmat	Dense matrix of REAL type	30
dSTRmat	Structure matrix of REAL type	31
dvector	Vector with n entries of REAL type	31
grid2d	Two dimensional grid data structure	32

iCOOmat	Sparse matrix of INT type in COO (or IJ) format	34
iCSRmat	Sparse matrix of INT type in CSR format	35
idenmat	Dense matrix of INT type	36
ILU_data	Data for ILU setup	36
ILU_param	Parameters for ILU	37
input_param	Input parameters	37
itsolver_param	Parameters passed to iterative solvers	44
ivector	Vector with n entries of INT type	45
Link	Struct for Links	46
linked_list	A linked list node	46
mxv_matfree	Matrix-vector multiplication, replace the actual matrix	47
precond	Preconditioner data and action	48
precond_block_data	Data passed to the preconditioner for block diagonal preconditioning	48
precond_block_reservoir_data	Data passed to the preconditioner for preconditioning reservoir simulation problems	49
precond_data	Data passed to the preconditioners	53
precond_data_bsr	Data passed to the preconditioners	54
precond_data_str	Data passed to the preconditioner for dSTRmat matrices	56
precond_diagbsr	Data passed to diagonal preconditioner for dBSRmat matrices	57
precond_diagstr	Data passed to diagonal preconditioner for dSTRmat matrices	58
precond_FASP_blkoi_data	Data passed to the preconditioner for preconditioning reservoir simulation problems	58
Schwarz_data	Data for Schwarz methods	62
Schwarz_param	Parameters for Schwarz method	63

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)	65
amg_setup_aggregation_bsr.inl	Utilities for multigrid cycles in BSR format	66
amg_setup_aggregation_csr.inl	Utilities for multigrid cycles for CSR matrices	66
amg_setup_cr.c	Brannick-Falgout compatible relaxation based AMG: SETUP phase	66
amg_setup_rs.c	Ruge-Stuben AMG: SETUP phase	67
amg_setup_sa.c	Smoothed aggregation AMG: SETUP phase	69
amg_setup_ua.c	Unsmoothed aggregation AMG: SETUP phase	70
amg_solve.c	Algebraic multigrid iterations: SOLVE phase	71
amlirecur.c	Abstract AMLI multilevel iteration – recursive version	74
array.c	Array operations	77
blas_array.c	BLAS operations for arrays	81
blas_bcsr.c	BLAS operations for block_dCSRmat matrices	86
blas_bsr.c	BLAS operations for dBSRmat matrices	89
blas_csr.c	BLAS operations for dCSRmat matrices	94
blas_csrl.c	BLAS operations for dCSRLmat matrices	102
blas_smat.c	BLAS operations for small full matrix	103
blas_str.c	BLAS operations for dSTRmat matrices	122

blas_vec.c	BLAS operations for vectors	124
checkmat.c	Check matrix properties	128
coarsening_cr.c	Coarsening with Brannick-Falgout strategy	133
coarsening_rs.c	Coarsening with a modified Ruge-Stuben strategy	134
convert.c	Some utilities for format conversion	136
doxygen.h	Main page for Doxygen documentation	140
eigen.c	Simple subroutines for compute the extreme eigenvalues	140
factor.f	LU factorization for CSR matrix	141
famg.c	Full AMG method as an iterative solver (main file)	141
fasp.h	Main header file for FASP	142
fasp_block.h	Main header file for FASP (block matrices)	151
fmgcycle.c	Abstract non-recursive full multigrid cycle	153
formats.c	Matrix format conversion routines	154
givens.c	Givens transformation	159
gmg_poisson.c	GMG method as an iterative solver for Poisson Problem	160
gmg_util.inl	Routines for GMG solvers	165
graphics.c	Functions for graphical output	165
ilu.f	ILU routines for preconditioning adapted from SPARSEKIT	167
ilu_setup_bsr.c	Setup Incomplete LU decomposition for dBSRmat matrices	168
ilu_setup_csr.c	Setup of ILU decomposition for dCSRmat matrices	169
ilu_setup_str.c	Setup of ILU decomposition for dSTRmat matrices	169
init.c	Initialize important data structures	171
input.c	Read input parameters	175
interface_mumps.c	Interface to MUMPS direct solvers	176
interface_samg.c	Interface to SAMG	178
interface_superlu.c	Interface to SuperLU direct solvers	180
interface_umfpack.c	Interface to UMFPACK direct solvers	181

interpolation.c	Interpolation operators for AMG	182
interpolation_em.c	Interpolation operators for AMG based on energy-min	184
io.c	Matrix-vector input/output subroutines	185
itsolver_bcsr.c	Iterative solvers for block_dCSRmat matrices	205
itsolver_bsr.c	Iterative solvers for dBSRmat matrices	206
itsolver_csr.c	Iterative solvers for dCSRmat matrices	210
itsolver_mf.c	Iterative solvers with matrix-free spmv	215
itsolver_str.c	Iterative solvers for dSTRmat matrices	218
itsolver_util.inl	Routines for iterative solvers	221
linklist.inl	Utilities for link list data structure	222
lu.c	LU decomposition and direct solve for dense matrix	223
memory.c	Memory allocation and deallocation	224
message.c	Output some useful messages	228
messages.h	Definition of all kinds of messages, including error messages, solver types, etc	231
mg_util.inl	Routines for algebraic multigrid cycles	248
mgcycle.c	Abstract non-recursive multigrid cycle	248
mgrecur.c	Abstract multigrid cycle – recursive version	249
ordering.c	A collection of ordering, merging, removing duplicated integers functions	250
parameters.c	Initialize, set, or print input data and parameters	255
pbcgs.c	Krylov subspace methods – Preconditioned BiCGstab	265
pbcgs_mf.c	Krylov subspace methods – Preconditioned BiCGstab (matrix free)	269
pcg.c	Krylov subspace methods – Preconditioned conjugate gradient	271
pcg_mf.c	Krylov subspace methods – Preconditioned conjugate gradient (matrix free)	275
pgcg.c	Krylov subspace methods – Preconditioned Generalized CG	277
pgcg_mf.c	Krylov subspace methods – Preconditioned Generalized CG (matrix free)	278
pgmres.c	Krylov subspace methods – Preconditioned GMRes	280
pgmres_mf.c	Krylov subspace methods – Preconditioned GMRes (matrix free)	283

pminres.c	Krylov subspace methods – Preconditioned minimal residual	284
pminres_mf.c	Krylov subspace methods – Preconditioned minimal residual (matrix free)	288
precond_bsr.c	Preconditioners for dBSRmat matrices	290
precond_csr.c	Preconditioners for dCSRmat matrices	296
precond_str.c	Preconditioners for dSTRmat matrices	302
pvfgmres.c	Krylov subspace methods – Preconditioned variable-restarting flexible GMRes	308
pvfgmres_mf.c	Krylov subspace methods – Preconditioned variable-restarting flexible GMRes (matrix free)	311
pvgmres.c	Krylov subspace methods – Preconditioned variable-restart GMRes	312
pvgmres_mf.c	Krylov subspace methods – Preconditioned variable-restarting GMRes (matrix free)	316
quadrature.c	Quadrature rules	318
rap.c	R*A*P driver	319
schwarz.f	Schwarz smoothers	320
schwarz_setup.c	Setup phase for the Schwarz methods	321
smat.c	Simple operations for <i>small</i> full matrices in row-major format	322
smoother_bsr.c	Smoothers for dBSRmat matrices	328
smoother_csr.c	Smoothers for dCSRmat matrices	338
smoother_csr_cr.c	Smoothers for dCSRmat matrices using compatible relaxation	343
smoother_csr_poly.c	Smoothers for dCSRmat matrices using poly. approx. to A^{-1}	345
smoother_str.c	Smoothers for dSTRmat matrices	346
sparse_block.c	Sparse matrix block operations	354
sparse_bsr.c	Sparse matrix operations for dBSRmat matrices	356
sparse_coo.c	Sparse matrix operations for dCOOmat matrices	363
sparse_csr.c	Sparse matrix operations for dCSRmat matrices	365
sparse_csrl.c	Sparse matrix operations for dCSRLmat matrices	379
sparse_str.c	Sparse matrix operations for dSTRmat matrices	380
sparse_util.c	Routines for sparse matrix operations	383
spbcgs.c	Krylov subspace methods – Preconditioned BiCGstab with safe net	391

spcg.c	Krylov subspace methods – Preconditioned conjugate gradient with safe net	396
spgmr.c	Krylov subspace methods – Preconditioned GMRes with safe net	399
spminr.c	Krylov subspace methods – Preconditioned minimal residual with safe net	402
spvgmr.c	Krylov subspace methods – Preconditioned variable-restart GMRes with safe net	406
threads.c	Get and set number of threads and assigne work load for each thread	409
timing.c	Timing subroutines	410
vec.c	Simple operations for vectors	411
wrapper.c	Wrappers for accessing functions by advanced users	418

Chapter 8

Data Structure Documentation

8.1 AMG_data Struct Reference

Data for AMG solvers.

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

Data Fields

- [SHORT max_levels](#)
max number of levels
- [SHORT num_levels](#)
number of levels in use \leq max_levels
- [dCSRmat A](#)
pointer to the matrix at level level_num
- [dCSRmat R](#)
restriction operator at level level_num
- [dCSRmat P](#)
prolongation operator at level level_num
- [dvector b](#)
pointer to the right-hand side at level level_num
- [dvector x](#)
pointer to the iterative solution at level level_num
- [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.

8.1.1 Detailed Description

Data for AMG solvers.

Note

This is needed for the AMG solver/preconditioner.

Definition at line 636 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>
```

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 diagin](#)

pointer to the diagonal inverse at level level_num
- [dCSRmat Ac](#)

pointer to the matrix at level level_num (csr format)
- [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 diaginvs_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
- [dvector w](#)
temporary work space

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 coarsest level dof
- **SHORT cycle_type**
type of AMG cycle
- **SHORT smoother**
smoother type
- **SHORT smooth_order**
smoother order
- **SHORT presmooth_iter**
number of presmootherers
- **SHORT postsmooth_iter**
number of postsmootherers
- **REAL relaxation**
relaxation parameter for SOR smoother
- **SHORT polynomial_degree**
degree of the polynomial smoother
- **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 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**
numebr of paths use to determin stongly 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_ifil](#)
level of fill-in for ILUs and ILUK
- [REAL ILU_droptol](#)
drop tolerance for ILU_t
- [REAL ILU_relax](#)
relaxation for ILUs
- [REAL ILU_permtol](#)
*permuted if $\text{permtol} * |a(i,j)| > |a(i,i)|$*
- [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

8.3.1 Detailed Description

Parameters for AMG solver.

Note

This is needed for the AMG solver/preconditioner.

Definition at line 512 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>
```

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>
```

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>
```

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>
```

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>
```

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>
```

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 dBSRmat Struct Reference

Block sparse row storage matrix of REAL type.

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

Data Fields

- [INT ROW](#)
number of rows of sub-blocks in matrix A, M
- [INT COL](#)
number of cols of sub-blocks in matrix A, N
- [INT NNZ](#)
number of nonzero sub-blocks in matrix A, NNZ
- [INT nb](#)
dimension of each sub-block
- [INT storage_manner](#)
storage manner for each sub-block
- [REAL * val](#)
- [INT * IA](#)
integer array of row pointers, the size is ROW+1
- [INT * JA](#)

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 * I](#)
integer array of row indices, the size is nnz
- [INT * J](#)
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.

Definition at line 215 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](#)
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 271 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 155 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 115 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(nc^2))*
- [INT nband](#)
number of off-diag bands
- [INT * offsets](#)
offsets of the off-diagals (length is nband)
- [REAL ** offdiag](#)
*off-diagonal entries (dimension is nband * [(ngrid-offsets) * nc^2])*

8.15.1 Detailed Description

Structure matrix of REAL type.

Note

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

Definition at line 310 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 348 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 1013 of file fasp.h.

8.17.2 Field Documentation

8.17.2.1 `INT(* e)[2]`

Vertices of edges

Definition at line 1016 of file fasp.h.

8.17.2.2 `INT edges`

Number of edges

Definition at line 1027 of file fasp.h.

8.17.2.3 `INT* ediri`

Boundary flags (0 <=> interior edge)

Definition at line 1020 of file fasp.h.

8.17.2.4 `INT* efather`

Father edge or triangle

Definition at line 1023 of file fasp.h.

8.17.2.5 `REAL(* p)[2]`

Coordinates of vertices

Definition at line 1015 of file fasp.h.

8.17.2.6 `INT* pdiri`

Boundary flags (0 <=> interior point)

Definition at line 1019 of file fasp.h.

8.17.2.7 `INT* pfather`

Father point or edge

Definition at line 1022 of file fasp.h.

8.17.2.8 `INT(* s)[3]`

Edges of triangles

Definition at line 1018 of file fasp.h.

8.17.2.9 INT(* t)[3]

Vertices of triangles

Definition at line 1017 of file fasp.h.

8.17.2.10 INT* tfather

Father triangle

Definition at line 1024 of file fasp.h.

8.17.2.11 INT triangles

Number of triangles

Definition at line 1028 of file fasp.h.

8.17.2.12 INT vertices

Number of grid points

Definition at line 1026 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 245 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 185 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 134 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 * ijl](#)
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 406 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 leve
- [SHORT ILU_type](#)
ILU type for decomposition.
- [INT ILU_lfil](#)
level of fill-in for ILUk
- [REAL ILU_droptol](#)
drop tolerance for ILU_t
- [REAL ILU_relax](#)
add the sum of dropped elements to diagnal element in proportion relax
- [REAL ILU_permtol](#)
*permuted if $\text{permtol} * |a(i,j)| > |a(i,i)|$*

8.22.1 Detailed Description

Parameters for ILU.

Definition at line 380 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 inifile [256]`
- `char workdir [256]`
- `INT problem_num`
- `SHORT solver_type`
- `SHORT precondition_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`
- `SHORT AMG_type`
- `SHORT AMG_levels`
- `SHORT AMG_cycle_type`
- `SHORT AMG_smoother`
- `SHORT AMG_smooth_order`
- `REAL AMG_relaxation`
- `SHORT AMG_polynomial_degree`
- `SHORT AMG_presmooth_iter`
- `SHORT AMG_postsMOOTH_iter`
- `INT AMG_coarse_dof`
- `REAL AMG_tol`
- `INT AMG_maxit`
- `SHORT AMG_ILU_levels`
- `SHORT AMG_coarse_scaling`
- `SHORT AMG_amli_degree`
- `SHORT AMG_nl_amli_krylov_type`
- `INT AMG_schwarz_levels`
- `SHORT AMG_coarsening_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`
- `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 920 of file fasp.h.

8.23.2 Field Documentation

8.23.2.1 INT AMG_aggressive_level

number of levels use aggressive coarsening

Definition at line 976 of file fasp.h.

8.23.2.2 INT AMG_aggressive_path

number of paths used to determine strongly coupled C-set

Definition at line 977 of file fasp.h.

8.23.2.3 SHORT AMG_amli_degree

degree of the polynomial used by AMLI cycle

Definition at line 966 of file fasp.h.

8.23.2.4 INT AMG_coarse_dof

minimal coarsest level dof

Definition at line 961 of file fasp.h.

8.23.2.5 SHORT AMG_coarse_scaling

switch of scaling of the coarse grid correction

Definition at line 965 of file fasp.h.

8.23.2.6 SHORT AMG_coarsening_type

coarsening type

Definition at line 971 of file fasp.h.

8.23.2.7 SHORT AMG_cycle_type

type of cycle

Definition at line 954 of file fasp.h.

8.23.2.8 SHORT AMG_ILU_levels

how many levels use ILU smoother

Definition at line 964 of file fasp.h.

8.23.2.9 SHORT AMG_interpolation_type

interpolation type

Definition at line 972 of file fasp.h.

8.23.2.10 SHORT AMG_levels

maximal number of levels

Definition at line 953 of file fasp.h.

8.23.2.11 INT AMG_max_aggregation

max size of each aggregate

Definition at line 981 of file fasp.h.

8.23.2.12 REAL AMG_max_row_sum

maximal row sum

Definition at line 975 of file fasp.h.

8.23.2.13 INT AMG_maxit

number of iterations for AMG used as preconditioner

Definition at line 963 of file fasp.h.

8.23.2.14 SHORT AMG_nl_amli_krylov_type

type of krylov method used by nonlinear AMLI cycle

Definition at line 967 of file fasp.h.

8.23.2.15 SHORT AMG_polynomial_degree

degree of the polynomial smoother

Definition at line 958 of file fasp.h.

8.23.2.16 SHORT AMG_postsmooth_iter

number of postsmoothing

Definition at line 960 of file fasp.h.

8.23.2.17 SHORT AMG_presmooth_iter

number of presmoothing

Definition at line 959 of file fasp.h.

8.23.2.18 REAL AMG_relaxation

over-relaxation parameter for SOR

Definition at line 957 of file fasp.h.

8.23.2.19 INT AMG_schwarz_levels

number of levels use schwarz smoother

Definition at line 968 of file fasp.h.

8.23.2.20 SHORT AMG_smooth_filter

use filter for smoothing the tentative prolongation or not

Definition at line 983 of file fasp.h.

8.23.2.21 SHORT AMG_smooth_order

order for smoothers

Definition at line 956 of file fasp.h.

8.23.2.22 SHORT AMG_smoother

type of smoother

Definition at line 955 of file fasp.h.

8.23.2.23 REAL AMG_strong_coupled

strong coupled threshold for aggregate

Definition at line 980 of file fasp.h.

8.23.2.24 REAL AMG_strong_threshold

strong threshold for coarsening

Definition at line 973 of file fasp.h.

8.23.2.25 REAL AMG_tentative_smooth

relaxation factor for smoothing the tentative prolongation

Definition at line 982 of file fasp.h.

8.23.2.26 REAL AMG_tol

tolerance for AMG if used as preconditioner

Definition at line 962 of file fasp.h.

8.23.2.27 REAL AMG_truncation_threshold

truncation factor for interpolation

Definition at line 974 of file fasp.h.

8.23.2.28 SHORT AMG_type

Type of AMG

Definition at line 952 of file fasp.h.

8.23.2.29 REAL ILU_droptol

drop tolerance

Definition at line 942 of file fasp.h.

8.23.2.30 INT ILU_lfil

level of fill-in

Definition at line 941 of file fasp.h.

8.23.2.31 REAL ILU_permtol

permutation tolerance

Definition at line 944 of file fasp.h.

8.23.2.32 REAL ILU_relax

scaling factor: add the sum of dropped entries to diagonal

Definition at line 943 of file fasp.h.

8.23.2.33 SHORT ILU_type

ILU type for decomposition

Definition at line 940 of file fasp.h.

8.23.2.34 char inifile[256]

ini file name

Definition at line 927 of file fasp.h.

8.23.2.35 INT itsolver_maxit

maximal number of iterations for iterative solvers

Definition at line 936 of file fasp.h.

8.23.2.36 REAL itsolver_tol

tolerance for iterative linear solver

Definition at line 935 of file fasp.h.

8.23.2.37 SHORT output_type

type of output stream

Definition at line 924 of file fasp.h.

8.23.2.38 SHORT precondition_type

type of preconditioner for iterative solvers

Definition at line 933 of file fasp.h.

8.23.2.39 SHORT print_level

print level

Definition at line 923 of file fasp.h.

8.23.2.40 INT problem_num

problem number to solve

Definition at line 929 of file fasp.h.

8.23.2.41 INT restart

restart number used in GMRES

Definition at line 937 of file fasp.h.

8.23.2.42 INT Schwarz_maxlvl

maximal levels

Definition at line 948 of file fasp.h.

8.23.2.43 INT Schwarz_mmsize

maximal block size

Definition at line 947 of file fasp.h.

8.23.2.44 INT Schwarz_type

type of schwarz method

Definition at line 949 of file fasp.h.

8.23.2.45 SHORT solver_type

type of iterative solvers

Definition at line 932 of file fasp.h.

8.23.2.46 SHORT stop_type

type of stopping criteria for iterative solvers

Definition at line 934 of file fasp.h.

8.23.2.47 char workdir[256]

working directory for data files

Definition at line 928 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 precondition_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 991 of file fasp.h.

8.24.2 Field Documentation

8.24.2.1 **SHORT** itsolver_type

solver type: see message.h

Definition at line 993 of file fasp.h.

8.24.2.2 **INT** maxit

max number of iterations

Definition at line 996 of file fasp.h.

8.24.2.3 **SHORT** precondition_type

preconditioner type: see message.h

Definition at line 994 of file fasp.h.

8.24.2.4 **SHORT** print_level

print level: 0–10

Definition at line 999 of file fasp.h.

8.24.2.5 **INT** restart

number of steps for restarting: for GMRES etc

Definition at line 998 of file fasp.h.

8.24.2.6 **SHORT** stop_type

stopping criteria type

Definition at line 995 of file fasp.h.

8.24.2.7 **REAL** tol

convergence tolerance

Definition at line 997 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 362 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 1040 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>
```

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 1057 of file fasp.h.

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

- [fasp.h](#)

8.28 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

8.28.1 Detailed Description

Matrix-vector multiplication, replace the actual matrix.

Definition at line 904 of file fasp.h.

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

- [fasp.h](#)

8.29 precondition 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

8.29.1 Detailed Description

Preconditioner data and action.

Note

This is the preconditioner structure for preconditioned iterative methods.

Definition at line 885 of file fasp.h.

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

- [fasp.h](#)

8.30 precondition_block_data Struct Reference

Data passed to the preconditioner for block diagonal preconditioning.

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

Data Fields

- [dCSRmat](#) * [A](#)
- [dvector](#) * [r](#)
- [dCSRmat](#) ** [Ablock](#)
- [ivector](#) ** [row_idx](#)
- [ivector](#) ** [col_idx](#)
- [AMG_param](#) * [amgparam](#)
- [dCSRmat](#) ** [Aarray](#)

8.30.1 Detailed Description

Data passed to the preconditioner for block diagonal preconditioning.

Note

This is needed for the diagonal block preconditioner.

Definition at line 460 of file fasp_block.h.

8.30.2 Field Documentation

8.30.2.1 dCSRmat* A

problem data, the sparse matrix

Definition at line 462 of file fasp_block.h.

8.30.2.2 dCSRmat** Aarray

data generated in the setup phase

Definition at line 470 of file fasp_block.h.

8.30.2.3 dCSRmat** Ablock

problem data, the blocks

Definition at line 465 of file fasp_block.h.

8.30.2.4 AMG_param* amgparam

parameters for AMG

Definition at line 469 of file fasp_block.h.

8.30.2.5 ivector** col_idx

problem data, col indices

Definition at line 467 of file fasp_block.h.

8.30.2.6 dvector* r

problem data, the right-hand side vector

Definition at line 463 of file fasp_block.h.

8.30.2.7 ivector** row_idx

problem data, row indices

Definition at line 466 of file fasp_block.h.

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

- [fasp_block.h](#)

8.31 precondition_block_reservoir_data Struct Reference

Data passed to the preconditioner for preconditioning reservoir simulation problems.

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

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_ifil](#)
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 $(-I - Awr*Arr^{-1}*Arw)^{-1}$, Arr may be replaced by LU*
- [dvector](#) * [DPSinvDSS](#)

$Diag(PS) * inv(Diag(SS))$

- [SHORT](#) scaled
- [ivector](#) * [perf_idx](#)
- [dSTRmat](#) * [RR](#)
- [dCSRmat](#) * [WW](#)
- [dCSRmat](#) * [PP](#)
- [dSTRmat](#) * [SS](#)
- [precond_diagstr](#) * [diag](#)
- [dvector](#) * [diaginv](#)
- [ivector](#) * [pivot](#)
- [dvector](#) * [diaginvS](#)
- [ivector](#) * [pivotS](#)
- [ivector](#) * [order](#)
- [dvector](#) [r](#)
- [REAL](#) * [w](#)

8.31.1 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 361 of file fasp_block.h.

8.31.2 Field Documentation

8.31.2.1 [precond_diagstr](#)* [diag](#)

the diagonal inverse for diagonal scaling

Definition at line 441 of file fasp_block.h.

8.31.2.2 [dvector](#)* [diaginv](#)

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

Definition at line 442 of file fasp_block.h.

8.31.2.3 [dvector](#)* [diaginvS](#)

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

Definition at line 444 of file fasp_block.h.

8.31.2.4 [ivector](#)* [order](#)

order for smoothing

Definition at line 446 of file fasp_block.h.

8.31.2.5 ivector* perf_idx

variable index for perf

Definition at line 434 of file fasp_block.h.

8.31.2.6 ivector* pivot

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

Definition at line 443 of file fasp_block.h.

8.31.2.7 ivector* pivotS

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

Definition at line 445 of file fasp_block.h.

8.31.2.8 dCSRmat* PP

pressure block after diagonal scaling

Definition at line 438 of file fasp_block.h.

8.31.2.9 dvector r

temporary dvector used to store and restore the residual

Definition at line 449 of file fasp_block.h.

8.31.2.10 dSTRmat* RR

Diagonal scaled reservoir block

Definition at line 436 of file fasp_block.h.

8.31.2.11 SHORT scaled

whether the matrix is scaled

Definition at line 433 of file fasp_block.h.

8.31.2.12 dSTRmat* SS

saturation block after diagonal scaling

Definition at line 439 of file fasp_block.h.

8.31.2.13 REAL* w

temporary work space for other usage

Definition at line 450 of file fasp_block.h.

8.31.2.14 dCSRmat* WW

Argumented well block

Definition at line 437 of file fasp_block.h.

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

- [fasp_block.h](#)

8.32 precondition_data Struct Reference

Data passed to the preconditioners.

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

Data Fields

- [SHORT AMG_type](#)
type of AMG method
- [SHORT print_level](#)
print level in AMG preconditioner
- [INT maxit](#)
max number of iterations of AMG preconditioner
- [SHORT max_levels](#)
max number of AMG levels
- [REAL tol](#)
tolerance for AMG preconditioner
- [SHORT cycle_type](#)
AMG cycle type.
- [SHORT smoother](#)
AMG smoother type.
- [SHORT smooth_order](#)
AMG smoother ordering.
- [SHORT presmooth_iter](#)
number of presmoothing
- [SHORT postsmooth_iter](#)
number of postsmoothing
- [REAL relaxation](#)
coarsening type
- [SHORT polynomial_degree](#)
degree of the polynomial smoother
- [SHORT coarsening_type](#)
switch of scaling of the coarse grid correction
- [SHORT coarse_scaling](#)
relaxation parameter for SOR smoother
- [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.
- [dvector r](#)
temporary dvector used to store and restore the residual
- [REAL * w](#)
temporary work space for other usage

8.32.1 Detailed Description

Data passed to the preconditioners.

Definition at line 697 of file fasp.h.

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

- [fasp.h](#)

8.33 precondition_data_bsr Struct Reference

Data passed to the preconditioners.

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

Data Fields

- [SHORT AMG_type](#)
type of AMG method
- [SHORT print_level](#)
print level in AMG preconditioner
- [INT maxit](#)
max number of iterations of AMG preconditioner
- [INT max_levels](#)
max number of AMG levels
- [REAL tol](#)
tolerance for AMG preconditioner
- [SHORT cycle_type](#)
AMG cycle type.
- [SHORT smoother](#)

- AMG smoother type.*
- [SHORT smooth_order](#)
 - AMG smoother ordering.*
- [SHORT presmooth_iter](#)
 - number of presmoothing*
- [SHORT postsmooth_iter](#)
 - number of postsmoothing*
- [SHORT coarsening_type](#)
 - coarsening type*
- [REAL relaxation](#)
 - relaxation parameter for SOR smoother*
- [SHORT coarse_scaling](#)
 - switch of scaling of the coarse grid correction*
- [SHORT amli_degree](#)
 - degree of the polynomial used by AMLI cycle*
- [REAL * amli_coef](#)
 - coefficients of the polynomial used by AMLI cycle*
- [REAL tentative_smooth](#)
 - smooth factor for smoothing the tentative prolongation*
- [SHORT nl_amli_krylov_type](#)
 - type of krylov method used by Nonlinear AMLI cycle*
- [AMG_data_bsr * mgl_data](#)
 - AMG preconditioner data.*
- [AMG_data * pres_mgl_data](#)
 - AMG preconditioner data for pressure block.*
- [ILU_data * LU](#)
 - ILU preconditioner data (needed for CPR type preconditioner)*
- [dBSRmat * A](#)
 - Matrix data.*
- [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.

Note

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

Definition at line 282 of file fasp_block.h.

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

- [fasp_block.h](#)

8.34 precondition_data_str Struct Reference

Data passed to the preconditioner for dSTRmat matrices.

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

Data Fields

- [SHORT AMG_type](#)
type of AMG method
- [SHORT print_level](#)
print level in AMG preconditioner
- [INT maxit](#)
max number of iterations of AMG preconditioner
- [SHORT max_levels](#)
max number of AMG levels
- [REAL tol](#)
tolerance for AMG preconditioner
- [SHORT cycle_type](#)
AMG cycle type.
- [SHORT smoother](#)
AMG smoother type.
- [SHORT presmooth_iter](#)
number of presmoothing
- [SHORT postsmooth_iter](#)
number of postsmoothing
- [SHORT coarsening_type](#)
coarsening type
- [REAL relaxation](#)
relaxation parameter for SOR smoother
- [SHORT coarse_scaling](#)
switch of scaling of the coarse grid correction
- [AMG_data](#) * [mgl_data](#)
AMG preconditioner data.
- [ILU_data](#) * [LU](#)
ILU preconditioner data (needed for CPR type preconditioner)
- [SHORT scaled](#)
whether the matrix are scaled or not
- [dCSRmat](#) * [A](#)
the original CSR matrix
- [dSTRmat](#) * [A_str](#)
store the whole reservoir block in STR format
- [dSTRmat](#) * [SS_str](#)
store Saturation block in STR format
- [dvector](#) * [diaginv](#)
the inverse of the diagonals for GS/block GS smoother (whole reservoir matrix)
- [ivector](#) * [pivot](#)

- the pivot for the GS/block GS smoother (whole reservoir matrix)*
- [dvector](#) * [diaginvS](#)
 - the inverse of the diagonals for GS/block GS smoother (saturation block)*
- [ivector](#) * [pivotS](#)
 - the pivot for the GS/block GS smoother (saturation block)*
- [ivector](#) * [order](#)
 - order for smoothing*
- [ivector](#) * [neigh](#)
 - array to store neighbor information*
- [dvector](#) [r](#)
 - temporary dvector used to store and restore the residual*
- [REAL](#) * [w](#)
 - temporary work space for other usage*

8.34.1 Detailed Description

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

Definition at line 776 of file [fasp.h](#).

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

- [fasp.h](#)

8.35 precondition_diagbsr Struct Reference

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

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

Data Fields

- [INT](#) [nb](#)
 - dimension of each sub-block*
- [dvector](#) [diag](#)
 - diagonal elements*

8.35.1 Detailed Description

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

Note

This is needed for the diagonal preconditioner.

Definition at line 264 of file [fasp_block.h](#).

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

- [fasp_block.h](#)

8.36 precondition_diagstr Struct Reference

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

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

Data Fields

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

8.36.1 Detailed Description

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

Note

This is needed for the diagonal preconditioner.

Definition at line 869 of file `fasp.h`.

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

- [fasp.h](#)

8.37 precondition_FASP_blkoi_data Struct Reference

Data passed to the preconditioner for preconditioning reservoir simulation problems.

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

Data Fields

- [block_BSR](#) * `A`
Part 1: Basic data.
- [SHORT](#) `scaled`
Part 2: Data for CPR-like preconditioner for reservoir block.
- [dvector](#) * `diaginv_noscale`
- [dBSRmat](#) * `RR`
- [ivector](#) * `neigh`
- [ivector](#) * `order`
- [dBSRmat](#) * `SS`
- [dvector](#) * `diaginv_S`
- [ivector](#) * `pivot_S`
- [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
- [REAL relaxation](#)
relaxation parameter for SOR smoother
- [SHORT coarse_scaling](#)
switch of scaling of coarse grid correction
- [SHORT aml_i_degree](#)
degree of the polynomial used by AMLI cycle
- [REAL * aml_i_coef](#)
coefficients of the polynomial used by AMLI cycle
- [REAL tentative_smooth](#)
relaxation parameter for smoothing the tentative prolongation
- [dvector * diaginv](#)
- [ivector * pivot](#)
- [ILU_data * LU](#)
data of ILU for reservoir block
- [ivector * perf_idx](#)
- [ivector * perf_neigh](#)
- [dCSRmat * WW](#)
- [void * Numeric](#)
data for direct solver for argumented well block
- [REAL * invS](#)
*inverse of the schur complement $(-I - Awr*Arr^{-1}*Arw)^{-1}$, Arr may be replaced by LU*
- [INT maxit](#)
- [INT restart](#)
- [REAL tol](#)
- [dvector r](#)
- [REAL * w](#)

8.37.1 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 480 of file fasp_block.h.

8.37.2 Field Documentation

8.37.2.1 **block_BSR*** A

Part 1: Basic data.

whole jacobian system in block_BSRmat

Definition at line 485 of file fasp_block.h.

8.37.2.2 **dvector*** diaginv

inverse of the diagonal blocks of reservoir block

Definition at line 554 of file fasp_block.h.

8.37.2.3 **dvector*** diaginv_noscale

inverse of diagonal blocks for diagonal scaling

Definition at line 492 of file fasp_block.h.

8.37.2.4 **dvector*** diaginv_S

inverse of the diagonal blocks of saturation block

Definition at line 501 of file fasp_block.h.

8.37.2.5 **INT** maxit

max number of iterations

Definition at line 572 of file fasp_block.h.

8.37.2.6 **AMG_data*** mgl_data

AMG data for presure-presure block

Definition at line 506 of file fasp_block.h.

8.37.2.7 ivector* neigh

neighbor information of the reservoir block

Definition at line 496 of file fasp_block.h.

8.37.2.8 ivector* order

ordering of the reservoir block

Definition at line 497 of file fasp_block.h.

8.37.2.9 ivector* perf_idx

index of blocks which have perforation

Definition at line 561 of file fasp_block.h.

8.37.2.10 ivector* perf_neigh

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

Definition at line 562 of file fasp_block.h.

8.37.2.11 ivector* pivot

pivot for the GS smoothers for the reservoir matrix

Definition at line 555 of file fasp_block.h.

8.37.2.12 ivector* pivot_S

pivoting for the GS smoothers for saturation block

Definition at line 502 of file fasp_block.h.

8.37.2.13 dCSRmat* PP

pressure block

Definition at line 505 of file fasp_block.h.

8.37.2.14 dvector r

temporary dvector used to store and restore the residual

Definition at line 577 of file fasp_block.h.

8.37.2.15 INT restart

number of iterations for restart

Definition at line 573 of file fasp_block.h.

8.37.2.16 dBSRmat* RR

reservoir block

Definition at line 493 of file fasp_block.h.

8.37.2.17 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 491 of file fasp_block.h.

8.37.2.18 dBSRmat* SS

saturation block

Definition at line 500 of file fasp_block.h.

8.37.2.19 REAL tol

tolerance

Definition at line 574 of file fasp_block.h.

8.37.2.20 REAL* w

temporary work space for other usage

Definition at line 578 of file fasp_block.h.

8.37.2.21 dCSRmat* WW

Argumented well block

Definition at line 563 of file fasp_block.h.

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

- [fasp_block.h](#)

8.38 Schwarz_data Struct Reference

Data for Schwarz methods.

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

Data Fields

- [dCSRmat A](#)
pointer to the matrix

- [INT nblk](#)
number of blocks
- [INT * iblock](#)
row index of blocks
- [INT * jblock](#)
column index of blocks
- [REAL * rhsloc](#)
local right hand side
- [REAL * au](#)
LU decomposition: the U block.
- [REAL * al](#)
LU decomposition: the L block.
- [INT schwarz_type](#)
Schwarz method type.
- [INT memt](#)
working space size
- [INT * mask](#)
mask
- [INT * maxa](#)
maxa

8.38.1 Detailed Description

Data for Schwarz methods.

This is needed for the schwarz solver, preconditioner/smoothen.

Definition at line 465 of file fasp.h.

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

- [fasp.h](#)

8.39 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](#)
maxiaml size of blocks

8.39.1 Detailed Description

Parameters for Schwarz method.

Added on 05/14/2012

Definition at line 441 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"
```

Functions

- void [fasp_solver_amg](#) ([dCSRmat](#) **A*, [dvector](#) **b*, [dvector](#) **x*, [AMG_param](#) **param*)
Solve $Ax = b$ by algebaric multigrid methods.

9.1.1 Detailed Description

AMG method as an iterative solver (main file)

Definition in file [amg.c](#).

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 algebaric 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 05/05/2013: Remove error handling from AMG setup

Definition at line 37 of file amg.c.

9.2 amg_setup_aggregation_bsr.inl File Reference

Utilities for multigrid cycles in BSR format.

9.2.1 Detailed Description

Utilities for multigrid cycles in BSR format.

Definition in file [amg_setup_aggregation_bsr.inl](#).

9.3 amg_setup_aggregation_csr.inl File Reference

Utilities for multigrid cycles for CSR matrices.

9.3.1 Detailed Description

Utilities for multigrid cycles for CSR matrices.

Definition in file [amg_setup_aggregation_csr.inl](#).

9.4 amg_setup_cr.c File Reference

Brannick-Falgout compatible relaxation based AMG: SETUP phase.

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

Functions

- [SHORT fasp_amg_setup_cr](#) ([AMG_data](#) *mgl, [AMG_param](#) *param)
Set up phase of Brannick Falgout CR coarsening for classic AMG.

9.4.1 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

Definition in file [amg_setup_cr.c](#).

9.4.2 Function Documentation

9.4.2.1 [SHORT fasp_amg_setup_cr](#) ([AMG_data](#) * *mgl*, [AMG_param](#) * *param*)

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

Parameters

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

Returns

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](#).

9.5 amg_setup_rs.c File Reference

Ruge-Stuben AMG: SETUP phase.

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

Functions

- [INT fasp_amg_setup_rs](#) ([AMG_data](#) *mgl, [AMG_param](#) *param)
Setup phase of Ruge and Stuben's classic AMG.
- [INT fasp_amg_setup_rs_omp](#) ([AMG_data](#) *mgl, [AMG_param](#) *param)
Setup of AMG based on R-S coarsening.

9.5.1 Detailed Description

Ruge-Stuben AMG: SETUP phase.

Note

Setup A, P, R, levels using classic AMG! Reffer to "Multigrid" by Stuben in 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.

Definition in file [amg_setup_rs.c](#).

9.5.2 Function Documentation

9.5.2.1 INT 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

SUCCESS if succeeded, 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.

Definition at line 47 of file [amg_setup_rs.c](#).

9.5.2.2 INT fasp_amg_setup_rs_omp ([AMG_data](#) * *mgl*, [AMG_param](#) * *param*)

Setup of AMG based on R-S coarsening.

Parameters

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

Returns

SUCCESS if succeeded, otherwise, error information.

Author

Chunsheng Feng, Xiaoqiang Yue

Date

03/11/2011

Definition at line 253 of file amg_setup_rs.c.

9.6 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 "amg_setup_aggregation_csr.inl"
```

Functions

- [SHORT fasp_amg_setup_sa](#) ([AMG_data](#) *mgl, [AMG_param](#) *param)
Set up phase of smoothed aggregation AMG.

9.6.1 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

Definition in file [amg_setup_sa.c](#).

9.6.2 Function Documentation

9.6.2.1 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

SUCCESS if succeed, error otherwise

Author

Xiaozhe Hu

Date

09/29/2009

Note

Only for testing smoothed P and unsmoothed A, not used usually.

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 50 of file `amg_setup_sa.c`.

9.7 `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 "amg_setup_aggregation_csr.inl"
#include "amg_setup_aggregation_bsr.inl"
```

Functions

- [SHORT fasp_amg_setup_ua](#) ([AMG_data](#) *mgl, [AMG_param](#) *param)
Set up phase of unsmoothed aggregation AMG.
- [SHORT fasp_amg_setup_ua_bsr](#) ([AMG_data_bsr](#) *mgl, [AMG_param](#) *param)
Set up phase of unsmoothed aggregation AMG (BSR format)

9.7.1 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

Definition in file [amg_setup_ua.c](#).

9.7.2 Function Documentation

9.7.2.1 `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

SUCCESS if succeed, error otherwise

Author

Xiaozhe Hu

Date

12/28/2011

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

9.7.2.2 `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

SUCCESS if succeed, error otherwise

Author

Xiaozhe Hu

Date

03/16/2012

Definition at line 70 of file `amg_setup_ua.c`.

9.8 amg_solve.c File Reference

Algebraic multigrid iterations: SOLVE phase.

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

Functions

- `INT fasp_amg_solve (AMG_data *mgl, AMG_param *param)`
AMG – SOLVE phase.
- `INT fasp_amg_solve_amli (AMG_data *mgl, AMG_param *param)`
AMLI – SOLVE phase.
- `INT fasp_amg_solve_nl_amli (AMG_data *mgl, AMG_param *param)`
Nonlinear AMLI — SOLVE phase.
- `void fasp_famg_solve (AMG_data *mgl, AMG_param *param)`
FMG – SOLVE phase.

9.8.1 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!

Definition in file [amg_solve.c](#).

9.8.2 Function Documentation

9.8.2.1 `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](#).

9.8.2.2 `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

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

Definition at line 119 of file amg_solve.c.

9.8.2.3 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

Definition at line 198 of file amg_solve.c.

9.8.2.4 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 270 of file amg_solve.c.

9.9 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"
```

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.

9.9.1 Detailed Description

Abstract AMLI multilevel iteration – recursive version.

Note

Contains AMLI and nonlinear AMLI cycles

TODO: Need to add a non-recursive version! –Chensong

Definition in file [amlirecur.c](#).

9.9.2 Function Documentation

9.9.2.1 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 743 of file amlirecur.c.

9.9.2.2 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

Modified by Chensong Zhang on 02/27/2013: update direct solvers.

Definition at line 37 of file amlirecur.c.

9.9.2.3 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 numebr of levels

Author

Xiaozhe Hu

Date

04/06/2010

Modified by Chensong Zhang on 02/27/2013: update direct solvers.

Definition at line 292 of file amlirecur.c.

9.9.2.4 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 numebr of levels

Author

Xiaozhe Hu

Date

04/06/2010

Modified by Chensong Zhang on 02/27/2013: update direct solvers.

Definition at line 565 of file amlirecur.c.

9.10 array.c File Reference

Array operations.

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

Functions

- void [fasp_array_null](#) (REAL *x)
Initialize an array.
- void [fasp_array_set](#) (const INT n, REAL *x, const REAL val)
Set initial value for an array to be x=val.
- void [fasp_iarray_set](#) (const INT n, INT *x, const INT val)
Set initial value for an array to be x=val.
- void [fasp_array_cp](#) (const INT n, REAL *x, REAL *y)
Copy an array to the other y=x.
- void [fasp_iarray_cp](#) (const INT n, INT *x, INT *y)
Copy an array to the other y=x.

- void `fasp_array_cp_nc3` (`REAL *x`, `REAL *y`)
Copy an array to the other $y=x$, the length is 3.
- void `fasp_array_cp_nc5` (`REAL *x`, `REAL *y`)
Copy an array to the other $y=x$, the length is 5.
- void `fasp_array_cp_nc7` (`REAL *x`, `REAL *y`)
Copy an array to the other $y=x$, the length is 7.

9.10.1 Detailed Description

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

Definition in file [array.c](#).

9.10.2 Function Documentation

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

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

Parameters

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

Author

Chensong Zhang

Date

2010/04/03

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

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

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

Parameters

<code>x</code>	Pointer to the original vector
<code>y</code>	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.

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

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

Parameters

<i>x</i>	Pointer to the original vector
<i>y</i>	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.

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

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

Parameters

<i>x</i>	Pointer to the original vector
<i>y</i>	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.

9.10.2.5 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.

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

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

Parameters

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

Author

Chensong Zhang

Date

04/03/2010

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

05/23/2012

Definition at line 52 of file array.c.

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

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

Parameters

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

Author

Chunsheng Feng, Xiaoqiang Yue

Date

05/23/2012

Definition at line 192 of file array.c.

9.10.2.8 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.

9.11 blas_array.c File Reference

BLAS operations for arrays.

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

Functions

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

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

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

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

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

- `REAL fasp_blas_array_dotprod` (const INT *n*, REAL **x*, REAL **y*)
Inner product of two arrays (x,y)
- `REAL fasp_blas_array_norm1` (const INT *n*, REAL **x*)
L1 norm of array x.
- `REAL fasp_blas_array_norm2` (const INT *n*, REAL **x*)
L2 norm of array x.
- `REAL fasp_blas_array_norminf` (const INT *n*, REAL **x*)
Linf norm of array x.

9.11.1 Detailed Description

BLAS operations for arrays.

Definition in file [blas_array.c](#).

9.11.2 Function Documentation

9.11.2.1 `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](#).

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

$y = a * x + b * y$

Parameters

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

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

$$y = a * x + y$$

Parameters

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

Author

Chensong Zhang

Date

07/01/2009

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.

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

$$z = a * x + y$$

Parameters

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

Author

Chensong Zhang

Date

07/01/2009

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

Definition at line 167 of file blas_array.c.

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

Inner product of two arrays (x,y)

Parameters

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

Returns

Inner product (x,y)

Author

Chensong Zhang

Date

07/01/2009

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

Definition at line 267 of file blas_array.c.

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

L1 norm of array x.

Parameters

n	Number of variables
x	Pointer to x

Returns

L1 norm of x

Author

Chensong Zhang

Date

07/01/2009

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

Definition at line 308 of file blas_array.c.

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

L2 norm of array x.

Parameters

n	Number of variables
x	Pointer to x

Returns

L2 norm of x

Author

Chensong Zhang

Date

07/01/2009

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

Definition at line 348 of file blas_array.c.

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

Linf norm of array x.

Parameters

n	Number of variables
x	Pointer to x

Returns

L_{∞} 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.

9.12 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"
```

Functions

- void [fasp_blas_bdcsr_aApy](#) (const [REAL](#) alpha, [block_dCSRmat](#) *A, [REAL](#) *x, [REAL](#) *y)
*Matrix-vector multiplication $y = \alpha A * x + y$.*
- void [fasp_blas_bdcsr_mxv](#) ([block_dCSRmat](#) *A, [REAL](#) *x, [REAL](#) *y)
*Matrix-vector multiplication $y = A * x$.*
- void [fasp_blas_bdbsr_aApy](#) (const [REAL](#) alpha, [block_BSR](#) *A, [REAL](#) *x, [REAL](#) *y)
*Matrix-vector multiplication $y = \alpha A * x + y$.*
- void [fasp_blas_bdbsr_mxv](#) ([block_BSR](#) *A, [REAL](#) *x, [REAL](#) *y)
*Matrix-vector multiplication $y = A * x$.*

9.12.1 Detailed Description

BLAS operations for [block_dCSRmat](#) matrices.

Definition in file [blas_bcsr.c](#).

9.12.2 Function Documentation

9.12.2.1 void fasp_blas_bdbsr_aApy (const REAL *alpha*, block_BSR * *A*, REAL * *x*, REAL * *y*)

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

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 231 of file blas_bcsr.c.

9.12.2.2 void fasp_blas_bdbsr_mnv ([block_BSR](#) * A, REAL * x, REAL * y)

Matrix-vector multiplication $y = A*x$.

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 269 of file blas_bcsr.c.

9.12.2.3 void fasp_blas_bdcsr_aApy (const REAL *alpha*, [block_dCSRmat](#) * 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_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.

9.12.2.4 void fasp_blas_bdcsv (block_dCSRmat * A, REAL * x, REAL * y)

Matrix-vector multiplication $y = A*x$.

Parameters

A	Pointer to block_dCSRmat matrix A
x	Pointer to array x
y	Pointer to array y

Author

Chensong Zhang

Date

04/27/2013

Definition at line 130 of file blas_bcsr.c.

9.13 blas_bsr.c File Reference

BLAS operations for [dBSRmat](#) matrices.

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

Functions

- 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_aAxy](#) (const [REAL](#) alpha, [dBSRmat](#) *A, [REAL](#) *x, [REAL](#) *y)
*Compute $y := \alpha * A * x + y$.*
- void [fasp_blas_dbsr_aAxy_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_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!!*

9.13.1 Detailed Description

BLAS operations for [dBSRmat](#) matrices.

Definition in file [blas_bsr.c](#).

9.13.2 Function Documentation

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

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

Parameters

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

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 39 of file [blas_bsr.c](#).

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

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

Parameters

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

<i>y</i>	Pointer to the array y
----------	------------------------

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 320 of file blas_bsr.c.

9.13.2.3 void fasp_blas_dbsr_aApy_agg (const REAL *alpha*, dBSRmat * *A*, REAL * *x*, REAL * *y*)

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

Parameters

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

Author

Xiaozhe Hu

Date

01/02/2014

Note

Works for general nb (Xiaozhe)

Definition at line 595 of file blas_bsr.c.

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

Compute $y := A x$.

Parameters

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

Author

Zhiyang Zhou

Date

10/25/2010

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

05/23/2012

Note

Works for general nb (Xiaozhe)

Definition at line 906 of file blas_bsr.c.

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

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

Parameters

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

Author

Xiaozhe Hu

Date

01/02/2014

Note

Works for general nb (Xiaozhe)

Definition at line 2653 of file blas_bsr.c.

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

[dBSRmat](#) sparse matrix multiplication $B=R*A*P$

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 $R*A*P$ (output)

Author

Xiaozhe Hu, Chunsheng Feng, Zheng Li

Date

10/24/2012

Note

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

Definition at line 5029 of file blas_bsr.c.

9.13.2.7 void fasp_blas_dbsr_rap1 ([dBSRmat](#) * *R*, [dBSRmat](#) * *A*, [dBSRmat](#) * *P*, [dBSRmat](#) * *B*)

[dBSRmat](#) sparse matrix multiplication $B=R*A*P$

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 $R*A*P$ (output)

Author

Chunsheng Feng, Xiaoqiang Yue and Xiaozhe Hu

Date

08/08/2011

Note

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

Definition at line 4845 of file blas_bsr.c.

9.13.2.8 void fasp_blas_dbsr_rap_agg ([dBSRmat](#) * *R*, [dBSRmat](#) * *A*, [dBSRmat](#) * *P*, [dBSRmat](#) * *B*)

[dBSRmat](#) sparse matrix multiplication $B=R*A*P$, where small block matrices in *P* and *R* are identity matrices!!

Parameters

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

Author

Xiaozhe Hu

Date

10/24/2012

Note

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

Definition at line 5287 of file blas_bsr.c.

9.14 blas_csr.c File Reference

BLAS operations for [dCSRmat](#) matrices.

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

Functions

- [INT fasp_blas_dcsr_add](#) ([dCSRmat](#) *A, const [REAL](#) alpha, [dCSRmat](#) *B, const [REAL](#) beta, [dCSRmat](#) *C)
*compute $C = \alpha*A + \beta*B$ in CSR format*
- void [fasp_blas_dcsr_axm](#) ([dCSRmat](#) *A, const [REAL](#) alpha)
Multiply a sparse matrix A in CSR format by a scalar alpha.
- void [fasp_blas_dcsr_mxv](#) ([dCSRmat](#) *A, [REAL](#) *x, [REAL](#) *y)
*Matrix-vector multiplication $y = A*x$.*
- void [fasp_blas_dcsr_mxv_agg](#) ([dCSRmat](#) *A, [REAL](#) *x, [REAL](#) *y)
*Matrix-vector multiplication $y = A*x$, where the entries of A are all ones.*
- void [fasp_blas_dcsr_aAxy](#) (const [REAL](#) alpha, [dCSRmat](#) *A, [REAL](#) *x, [REAL](#) *y)
*Matrix-vector multiplication $y = \alpha*A*x + y$.*
- void [fasp_blas_dcsr_aAxy_agg](#) (const [REAL](#) alpha, [dCSRmat](#) *A, [REAL](#) *x, [REAL](#) *y)
*Matrix-vector multiplication $y = \alpha*A*x + y$, where the entries of A are all ones.*
- [REAL fasp_blas_dcsr_vmv](#) ([dCSRmat](#) *A, [REAL](#) *x, [REAL](#) *y)
*vector-Matrix-vector multiplication $\alpha = y'*A*x$*
- void [fasp_blas_dcsr_mxm](#) ([dCSRmat](#) *A, [dCSRmat](#) *B, [dCSRmat](#) *C)
*Sparse matrix multiplication $C=A*B$.*

- void [fasp_blas_dcsr_rap](#) ([dCSRmat](#) *R, [dCSRmat](#) *A, [dCSRmat](#) *P, [dCSRmat](#) *RAP)
*Triple sparse matrix multiplication $B=R*A*P$.*
- void [fasp_blas_dcsr_rap_agg](#) ([dCSRmat](#) *R, [dCSRmat](#) *A, [dCSRmat](#) *P, [dCSRmat](#) *RAP)
*Triple sparse matrix multiplication $B=R*A*P$.*
- void [fasp_blas_dcsr_rap_agg1](#) ([dCSRmat](#) *R, [dCSRmat](#) *A, [dCSRmat](#) *P, [dCSRmat](#) *B)
*Triple sparse matrix multiplication $B=R*A*P$, where the entries of R and P are all ones.*
- void [fasp_blas_dcsr_ptap](#) ([dCSRmat](#) *Pt, [dCSRmat](#) *A, [dCSRmat](#) *P, [dCSRmat](#) *Ac)
*Triple sparse matrix multiplication $B=P*A*P$.*
- void [fasp_blas_dcsr_rap4](#) ([dCSRmat](#) *R, [dCSRmat](#) *A, [dCSRmat](#) *P, [dCSRmat](#) *B, [INT](#) *icor_ysk)
*Triple sparse matrix multiplication $B=R*A*P$.*

9.14.1 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.

Definition in file [blas_csr.c](#).

9.14.2 Function Documentation

9.14.2.1 void fasp_blas_dcsr_aApy (const [REAL](#) *alpha*, [dCSRmat](#) * A, [REAL](#) * x, [REAL](#) * y)

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

Parameters

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

Author

Chensong Zhang

Date

07/01/2009

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

Definition at line 482 of file blas_csr.c.

9.14.2.2 void fasp_blas_dcsr_aApy_agg (const REAL *alpha*, dCSRmat * *A*, REAL * *x*, REAL * *y*)

Matrix-vector multiplication $y = \alpha * A * x + y$, where 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 596 of file blas_csr.c.

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

Returns

SUCCESS if succeeds, RUN_FAIL 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.

9.14.2.4 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 A
<i>alpha</i>	REAL factor alpha

Author

Chensong Zhang

Date

07/01/2009

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

Definition at line 203 of file blas_csr.c.

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

Author

Xiaozhe Hu

Date

11/07/2009

Note

This fct will be replaced! –Chensong

Definition at line 762 of file blas_csr.c.

9.14.2.6 void fasp_blas_dcsr_m xv ([dCSRmat](#) * *A*, REAL * *x*, REAL * *y*)

Matrix-vector multiplication $y = A*x$.

Parameters

<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 227 of file blas_csr.c.

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

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

<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 425 of file blas_csr.c.

9.14.2.8 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^*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 1600 of file blas_csr.c.

9.14.2.9 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 869 of file blas_csr.c.

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

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

Parameters

<i>R</i>	pointer to the dCSRmat matrix
<i>A</i>	pointer to the dCSRmat matrix

<i>P</i>	pointer to the dCSRmat matrix
<i>B</i>	pointer to dCSRmat matrix equal to $R*A*P$
<i>icor_ysk</i>	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 1699 of file blas_csr.c.

9.14.2.11 `void fasp_blas_dcsr_rap_agg (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

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.

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

Triple sparse matrix multiplication $B=R*A*P$, where the entries of R and P are all ones.

Parameters

R	Pointer to the dCSRmat matrix R
A	Pointer to the dCSRmat matrix A
P	Pointer to the dCSRmat matrix P
B	Pointer to dCSRmat matrix equal to $R*A*P$

Author

Xiaozhe Hu

Date

02/21/2011

Note

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

Definition at line 1417 of file blas_csr.c.

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

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

Parameters

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

Author

Chensong Zhang

Date

07/01/2009

Definition at line 707 of file blas_csr.c.

9.15 `blas_csrl.c` File Reference

BLAS operations for [dCSRLmat](#) matrices.

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

Functions

- void [fasp_blas_dcsrl_mxv](#) ([dCSRLmat](#) *A, [REAL](#) *x, [REAL](#) *y)
Compute $y = A*x$ for a sparse matrix in CSRL format.

9.15.1 Detailed Description

BLAS operations for [dCSRmat](#) matrices.

Note

For details of CSR 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.

Definition in file [blas_csrl.c](#).

9.15.2 Function Documentation

9.15.2.1 void fasp_blas_dcsrl_mnv (dCSRmat * A, REAL * x, REAL * y)

Compute $y = A \cdot x$ for a sparse matrix in CSR format.

Parameters

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

Date

2011/01/07

Definition at line 28 of file blas_csrl.c.

9.16 blas_smat.c File Reference

BLAS operations for small full matrix.

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

Functions

- void [fasp_blas_smat_mnv_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_mnv_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_mnv_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_mnv_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_mnv](#) (REAL *a, REAL *b, REAL *c, const INT n)
Compute the product of a small full matrix a and a array b, stored in c.
- void [fasp_blas_smat_mul_nc2](#) (REAL *a, REAL *b, REAL *c)

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

 - void `fasp_blas_smat_mul_nc3` (REAL *a, REAL *b, REAL *c)
- Compute the matrix product of two 3*3 matrices a and b, stored in c.*

 - void `fasp_blas_smat_mul_nc5` (REAL *a, REAL *b, REAL *c)
- Compute the matrix product of two 5*5 matrices a and b, stored in c.*

 - void `fasp_blas_smat_mul_nc7` (REAL *a, REAL *b, REAL *c)
- Compute the matrix product of two 7*7 matrices a and b, stored in c.*

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

 - void `fasp_blas_array_axpyz_nc2` (REAL a, REAL *x, REAL *y, REAL *z)

$z = a*x + y$

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

$z = a*x + y$

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

$z = a*x + y$

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

$z = a*x + y$

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

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

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

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

- void `fasp_blas_array_axpy_nc5` (const REAL a, REAL *x, REAL *y)

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

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

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

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

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

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

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

- void `fasp_blas_smat_ypAx_nc5` (REAL *A, REAL *x, REAL *y)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- void `fasp_blas_smat_ymAx_ns2` (`REAL *A`, `REAL *x`, `REAL *y`)
Compute $ys := ys - Ass * xs$, where 'A' is a 2*2 dense matrix, Ass is its saturaton part 1*1.
- void `fasp_blas_smat_ymAx_ns3` (`REAL *A`, `REAL *x`, `REAL *y`)
Compute $ys := ys - Ass * xs$, where 'A' is a 3*3 dense matrix, Ass is its saturaton part 2*2.
- void `fasp_blas_smat_ymAx_ns5` (`REAL *A`, `REAL *x`, `REAL *y`)
Compute $ys := ys - Ass * xs$, where 'A' is a 5*5 dense matrix, Ass is its saturaton part 4*4.
- void `fasp_blas_smat_ymAx_ns7` (`REAL *A`, `REAL *x`, `REAL *y`)
Compute $ys := ys - Ass * xs$, where 'A' is a 7*7 dense matrix, Ass is its saturaton part 6*6.
- void `fasp_blas_smat_ymAx_ns` (`REAL *A`, `REAL *x`, `REAL *y`, `const INT n`)
Compute $ys := ys - Ass * xs$, where 'A' is a $n*n$ dense matrix, Ass is its saturaton part $(n-1)*(n-1)$.

9.16.1 Detailed Description

BLAS operations for small full matrix.

Definition in file [blas_smat.c](#).

9.16.2 Function Documentation

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

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

Parameters

<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

Date

18/11/2011

Definition at line 627 of file blas_smat.c.

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

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

Parameters

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

y	Pointer to the destination array
-----	----------------------------------

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 650 of file blas_smat.c.

9.16.2.3 void fasp_blas_array_axpy_nc5 (const REAL a, REAL * x, REAL * y) $y = a*x + y$, the length of x and y is 5**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 679 of file blas_smat.c.

9.16.2.4 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**

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 726 of file blas_smat.c.

9.16.2.5 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 442 of file blas_smat.c.

9.16.2.6 void fasp_blas_array_axpyz_nc3 (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 3

Definition at line 469 of file blas_smat.c.

9.16.2.7 void fasp_blas_array_axpyz_nc5 (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 5

Definition at line 502 of file blas_smat.c.

9.16.2.8 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 553 of file blas_smat.c.

9.16.2.9 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 x and y

Author

Zhiyang Zhou

Date

2010/10/25

Definition at line 1250 of file blas_smat.c.

9.16.2.10 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 390 of file blas_smat.c.

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

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

Parameters

<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 175 of file blas_smat.c.

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

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

Parameters

<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 204 of file blas_smat.c.

9.16.2.13 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 241 of file blas_smat.c.

9.16.2.14 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 300 of file blas_smat.c.

9.16.2.15 void fasp_blas_smat_m xv (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 <i>n</i> * <i>n</i> matrix
<i>b</i>	Pointer to the REAL array with length <i>n</i>
<i>c</i>	Pointer to the REAL array with length <i>n</i>
<i>n</i>	Dimension of the matrix

Author

Xiaozhe Hu, Shiquan Zhang

Date

04/21/2010

Definition at line 125 of file blas_smat.c.

9.16.2.16 void fasp_blas_smat_m xv_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 25 of file blas_smat.c.

9.16.2.17 void fasp_blas_smat_mnv_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 47 of file blas_smat.c.

9.16.2.18 void fasp_blas_smat_mnv_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 70 of file blas_smat.c.

9.16.2.19 void fasp_blas_smat_mnv_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 96 of file blas_smat.c.

9.16.2.20 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

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

Author

Zhiyang Zhou, Xiaozhe Hu

Date

2010/10/25

Definition at line 1149 of file blas_smat.c.

9.16.2.21 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

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

Author

Xiaozhe Hu

Date

18/11/2011

Note

Works for 2-component

Definition at line 1019 of file blas_smat.c.

9.16.2.22 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 1047 of file blas_smat.c.

9.16.2.23 `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 1077 of file blas_smat.c.

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

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

Parameters

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

y	Pointer to the REAL array with length 7
-----	---

Author

Xiaozhe Hu, Zhiyang Zhou

Date

01/06/2011

Note

Works for 7-component

Definition at line 1111 of file blas_smat.c.

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

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

Parameters

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

Author

Xiaozhe Hu

Date

2010/10/25

Note

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

Definition at line 1424 of file blas_smat.c.

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

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

Parameters

A	Pointer to the 2×2 dense matrix
-----	--

x	Pointer to the REAL array with length 1
y	Pointer to the REAL array with length 1

Author

Xiaozhe Hu

Date

2011/11/18

Note

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

Definition at line 1300 of file blas_smat.c.

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

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

Parameters

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

Author

Xiaozhe Hu

Date

2010/10/25

Note

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

Definition at line 1324 of file blas_smat.c.

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

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

Parameters

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

Author

Xiaozhe Hu

Date

2010/10/25

Note

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

Definition at line 1352 of file blas_smat.c.

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

Compute $y_s := y_s - A_{ss}x_s$, where 'A' is a 7*7 dense matrix, A_{ss} is its saturation part 6*6.

Parameters

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

Author

Xiaozhe Hu

Date

2010/10/25

Note

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

Definition at line 1386 of file blas_smat.c.

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

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

Parameters

A	Pointer to the $n \times n$ dense matrix
x	Pointer to the REAL array with length n
y	Pointer to the REAL array with length n
n	Dimension of the dense matrix

Author

Zhiyang Zhou

Date

2010/10/25

Definition at line 918 of file blas_smat.c.

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

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

Parameters

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

Author

Xiaozhe Hu

Date

2011/11/18

Definition at line 799 of file blas_smat.c.

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

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

Parameters

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

Author

Zhiyang Zhou, Xiaozhe Hu

Date

2010/10/25

Definition at line 825 of file blas_smat.c.

9.16.2.33 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 852 of file blas_smat.c.

9.16.2.34 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 883 of file blas_smat.c.

9.17 blas_str.c File Reference

BLAS operations for [dSTRmat](#) matrices.

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

Functions

- void [fasp_blas_dstr_aAxy](#) (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)
 $B = D^{\{-1\}} A$.

9.17.1 Detailed Description

BLAS operations for [dSTRmat](#) matrices.

Definition in file [blas_str.c](#).

9.17.2 Function Documentation

9.17.2.1 void fasp_blas_dstr_aApy (REAL *alpha*, dSTRmat * *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 dSTRmat matrix
<i>x</i>	Pointer to REAL array
<i>y</i>	Pointer to REAL array

Author

Zhiyang Zhou, Xiaozhe Hu, Shiquan Zhang

Date

2010/10/15

Definition at line 47 of file blas_str.c.

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

Matrix-vector multiplication $y = A x$.

Parameters

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

Author

Chensong Zhang

Date

04/27/2013

Definition at line 117 of file blas_str.c.

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

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

Parameters

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

Author

Shiquan Zhang

Date

2010/10/15

Modified by Chunsheng Feng, Zheng Li

Date

08/30/2012

Definition at line 142 of file blas_str.c.

9.18 blas_vec.c File Reference

BLAS operations for vectors.

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

Functions

- void [fasp_blas_dvec_axpy](#) (const [REAL](#) a, [dvector](#) *x, [dvector](#) *y)
 $y = a*x + y$
- void [fasp_blas_dvec_axpyz](#) (const [REAL](#) a, [dvector](#) *x, [dvector](#) *y, [dvector](#) *z)
 $z = a*x + y$, *z is a third vector (z is cleared)*
- [REAL fasp_blas_dvec_dotprod](#) ([dvector](#) *x, [dvector](#) *y)
Inner product of two vectors (x,y)
- [REAL fasp_blas_dvec_relerr](#) ([dvector](#) *x, [dvector](#) *y)
Relative error of two dvector x and y.
- [REAL fasp_blas_dvec_norm1](#) ([dvector](#) *x)
L1 norm of dvector x.
- [REAL fasp_blas_dvec_norm2](#) ([dvector](#) *x)
L2 norm of dvector x.
- [REAL fasp_blas_dvec_norminf](#) ([dvector](#) *x)
Linf norm of dvector x.

9.18.1 Detailed Description

BLAS operations for vectors.

Definition in file [blas_vec.c](#).

9.18.2 Function Documentation

9.18.2.1 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

07/01/2009

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

05/23/2012

Definition at line 35 of file blas_vec.c.

9.18.2.2 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

07/01/2009

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

05/23/2012

Definition at line 87 of file blas_vec.c.

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

Inner product of two vectors (x, y)

Parameters

x	Pointer to dvector x
y	Pointer to dvector y

Returns

Inner product

Author

Chensong Zhang

Date

07/01/209

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

05/23/2012

Definition at line 123 of file blas_vec.c.

9.18.2.4 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

07/01/209

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

05/23/2012

Definition at line 224 of file blas_vec.c.

9.18.2.5 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 267 of file blas_vec.c.

9.18.2.6 REAL fasp_blas_dvec_norminf (dvector * x)

Linf norm of dvector x .

Parameters

x	Pointer to dvector x
-----	------------------------

Returns

L_{∞} norm of x

Author

Chensong Zhang

Date

07/01/2009

Definition at line 307 of file blas_vec.c.

9.18.2.7 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

07/01/209

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

05/23/2012

Definition at line 169 of file blas_vec.c.

9.19 checkmat.c File Reference

Check matrix properties.

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

Functions

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

9.19.1 Detailed Description

Check matrix properties.

Definition in file [checkmat.c](#).

9.19.2 Function Documentation

9.19.2.1 **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 276 of file checkmat.c.

9.19.2.2 **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 109 of file checkmat.c.

9.19.2.3 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.

9.19.2.4 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

SUCCESS if no diagonal entry is close to zero, else ERROR (negative value)

Author

Shuo Zhang

Date

03/29/2009

Definition at line 64 of file checkmat.c.

9.19.2.5 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 310 of file checkmat.c.

9.19.2.6 INT fasp_check_symm (dCSRmat * A)

Check symmetry of a sparse matrix of CSR format.

Parameters

<i>A</i>	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 154 of file checkmat.c.

9.20 coarsening_cr.c File Reference

Coarsening with Brannick-Falgout strategy.

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

Functions

- [INT fasp_amg_coarsening_cr](#) (INT *i_0*, INT *i_n*, [dCSRmat](#) **A*, [ivector](#) **vertices*, [AMG_param](#) **param*)
CR coarsening.

9.20.1 Detailed Description

Coarsening with Brannick-Falgout strategy.

Definition in file [coarsening_cr.c](#).

9.20.2 Function Documentation

9.20.2.1 [INT fasp_amg_coarsening_cr](#) (INT *i_0*, INT *i_n*, [dCSRmat](#) * *A*, [ivector](#) * *vertices*, [AMG_param](#) * *param*)

CR coarsening.

Parameters

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

9.21 coarsening_rs.c File Reference

Coarsening with a modified Ruge-Stuben strategy.

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

Functions

- [INT fasp_amg_coarsening_rs](#) ([dCSRmat](#) *A, [ivector](#) *vertices, [dCSRmat](#) *P, [iCSRmat](#) *S, [AMG_param](#) *param)

Standard and aggressive coarsening schemes.

9.21.1 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.

Definition in file [coarsening_rs.c](#).

9.21.2 Function Documentation

9.21.2.1 INT fasp_amg_coarsening_rs (dCSRmat * *A*, ivecator * *vertices*, dCSRmat * *P*, iCSRmat * *S*, AMG_param * *param*)

Standard and aggressive coarsening schemes.

Parameters

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

Returns

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 56 of file coarsening_rs.c.

9.22 `convert.c` File Reference

Some utilities for format conversion.

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

Functions

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

9.22.1 Detailed Description

Some utilities for format conversion.

Definition in file [convert.c](#).

9.22.2 Function Documentation

9.22.2.1 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 <i>endianflag</i> = 1, it returns <i>inum</i> itself If <i>endianflag</i> = 2, it returns the swapped <i>inum</i>

Returns

Value of *inum* or swapped *inum*

Author

Ziteng Wang

Date

2012-12-24

Definition at line 105 of file [convert.c](#).

9.22.2.2 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 <i>endianflag</i> = 1, it returns <i>rnum</i> itself If <i>endianflag</i> = 2, it returns the swapped <i>rnum</i>

Returns

Value of *rnum* or swapped *rnum*

Author

Ziteng Wang

Date

2012-12-24

Definition at line 137 of file [convert.c](#).

9.22.2.3 `double fasp_aux_bbyteToldouble (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.

9.22.2.4 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.

9.22.2.5 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.

9.23 doxygen.h File Reference

Main page for Doygen documentation.

9.23.1 Detailed Description

Main page for Doygen documentation.

Definition in file [doxygen.h](#).

9.24 eigen.c File Reference

Simple subroutines for compute the extreme eigenvalues.

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

Functions

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

9.24.1 Detailed Description

Simple subroutines for compute the extreme eigenvalues.

Definition in file [eigen.c](#).

9.24.2 Function Documentation

9.24.2.1 [REAL fasp_dcsr_eig](#) ([dCSRmat](#) * A, const [REAL](#) tol, const [INT](#) maxit)

Approximate the largest eigenvalue of A by the power method.

Parameters

<i>A</i>	Pointer to the dCSRmat 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.

9.25 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)

9.25.1 Detailed Description

LU factorization for CSR matrix.

Author

Ludmil Zikatanov

Date

01/01/2002

Definition in file [factor.f](#).

9.26 famg.c File Reference

full AMG method as an iterative solver (main file)

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

Functions

- void `fasp_solver_famg` (`dCSRmat *A`, `dvector *b`, `dvector *x`, `AMG_param *param`)
Solve $Ax=b$ by full AMG.

9.26.1 Detailed Description

full AMG method as an iterative solver (main file)

Definition in file `famg.c`.

9.26.2 Function Documentation

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

Solve $Ax=b$ by full AMG.

Parameters

<i>A</i>	Pointer to <code>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>param</i>	Pointer to <code>AMG_param</code> : 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`.

9.27 fasp.h File Reference

Main header file for FASP.

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

Data Structures

- struct `dmat`
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 [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 `DIAGONAL_PREF` OFF
 - Flags for developer's use only.*
- #define `FASP_USE_ILU` ON
 - For external software package support.*
- #define `DLMALLOC` OFF
- #define `NEDMALLOC` OFF
- #define `RS_C1` ON
 - For internal code branches and data types.*
- #define `SHORT` short
- #define `INT` int
- #define `LONG` long
- #define `LONGLONG` long long
- #define `REAL` double
- #define `BIGREAL` 1e+20
 - Some global constants.*
- #define `SMALLREAL` 1e-20
- #define `MAX_REFINE_LVL` 20
- #define `MAX_AMG_LVL` 20
- #define `STAG_RATIO` 1e-4
- #define `MAX_STAG` 20
- #define `MAX_RESTART` 20
- #define `OPENMP_HOLDS` 2000
- #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 `ISTART` 0
 - Index starting point: C convention or Fortran convention.*
- #define `N2C(ind)` ((ind)-ISTART)
- #define `C2N(ind)` ((ind)+ISTART)
- #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](#)
- INT [nx_rb](#)
- INT [ny_rb](#)
- INT [nz_rb](#)
- INT * [IMAP](#)
- INT [MAXIMAP](#)

9.27.1 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.

Definition in file [fasp.h](#).

9.27.2 Macro Definition Documentation

9.27.2.1 #define __FASP_HEADER__

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

Definition at line 23 of file [fasp.h](#).

9.27.2.2 #define ABS(a) (((a)>=0.0)?(a):-{a})

absolute value of a

Definition at line 80 of file [fasp.h](#).

9.27.2.3 #define BIGREAL 1e+20

Some global constants.

A large real number

Definition at line 66 of file [fasp.h](#).

9.27.2.4 #define C2N(ind) ((ind)+ISTART)

map from C index 0,1,... to Natural index 1,2,...

Definition at line 96 of file fasp.h.

9.27.2.5 #define DIAGONAL_PREF OFF

Flags for developer's use only.

Flags for diagonal preference data structures order each row such that diagonal appears first

Definition at line 42 of file fasp.h.

9.27.2.6 #define DLMALLOC OFF

use dmalloc instead of standard malloc

Definition at line 50 of file fasp.h.

9.27.2.7 #define FASP_GSRB 1

MG level 0 use RedBlack Gauss Seidel Smoothing

Definition at line 1085 of file fasp.h.

9.27.2.8 #define FASP_USE_ILU ON

For external software package support.

enable ILU or not

Definition at line 49 of file fasp.h.

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

is a >= b?

Definition at line 86 of file fasp.h.

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

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

is a > b?

Definition at line 85 of file fasp.h.

9.27.2.11 #define INT int

regular integer type: int or long

Definition at line 58 of file fasp.h.

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

is $a == \text{NAN}$?

Definition at line 89 of file fasp.h.

9.27.2.13 `#define ISTART 0`

Index starting point: C convention or Fortran convention.

0 if in Natural index, 1 if data is in C index

Definition at line 94 of file fasp.h.

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

is $a \leq b$?

Definition at line 88 of file fasp.h.

9.27.2.15 `#define LONG long`

long integer type

Definition at line 59 of file fasp.h.

9.27.2.16 `#define LONGLONG long long`

long integer type

Definition at line 60 of file fasp.h.

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

is $a < b$?

Definition at line 87 of file fasp.h.

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

Definition of max, min, abs.

bigger one in a and b

Definition at line 78 of file fasp.h.

9.27.2.19 `#define MAX_AMG_LVL 20`

Maximal AMG coarsening level

Definition at line 69 of file fasp.h.

9.27.2.20 #define MAX_REFINE_LVL 20

Maximal refinement level

Definition at line 68 of file fasp.h.

9.27.2.21 #define MAX_RESTART 20

Maximal number of restarting

Definition at line 72 of file fasp.h.

9.27.2.22 #define MAX_STAG 20

Maximal number of stagnation times

Definition at line 71 of file fasp.h.

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

smaller one in a and b

Definition at line 79 of file fasp.h.

9.27.2.24 #define N2C(ind) ((ind)-ISTART)

map from Natural index 1,2,... to C index 0,1,...

Definition at line 95 of file fasp.h.

9.27.2.25 #define NEDMALLOC OFF

use nedmalloc instead of standard malloc

Definition at line 51 of file fasp.h.

9.27.2.26 #define OPENMP_HOLDS 2000

Switch to sequence or openmp

Definition at line 73 of file fasp.h.

9.27.2.27 #define REAL double

float type

Definition at line 61 of file fasp.h.

9.27.2.28 #define RS_C1 ON

For internal code branches and data types.

CF splitting of RS: check C1 Criterion

Definition at line 56 of file fasp.h.

9.27.2.29 `#define SHORT short`

short integer type

Definition at line 57 of file fasp.h.

9.27.2.30 `#define SMALLREAL 1e-20`

A small real number

Definition at line 67 of file fasp.h.

9.27.2.31 `#define STAG_RATIO 1e-4`

Stagantion tolerance = $\text{tol} * \text{STAGRATIO}$

Definition at line 70 of file fasp.h.

9.27.3 Typedef Documentation

9.27.3.1 `typedef struct dCOOmat dCOOmat`

Sparse matrix of REAL type in COO format

9.27.3.2 `typedef struct dCSRLmat dCSRLmat`

Sparse matrix of REAL type in CSRL format

9.27.3.3 `typedef struct dCSRmat dCSRmat`

Sparse matrix of REAL type in CSR format

9.27.3.4 `typedef struct ddenmat ddenmat`

Dense matrix of REAL type

9.27.3.5 `typedef struct dSTRmat dSTRmat`

Structured matrix of REAL type

9.27.3.6 `typedef struct dvector dvector`

Vector of REAL type

9.27.3.7 `typedef struct grid2d grid2d`

2D grid type for plotting

9.27.3.8 `typedef struct iCOOmat iCOOmat`

Sparse matrix of INT type in COO format

9.27.3.9 `typedef struct iCSRmat iCSRmat`

Sparse matrix of INT type in CSR format

9.27.3.10 `typedef struct idenmat idenmat`

Dense matrix of INT type

9.27.3.11 `typedef struct ivector ivector`

Vector of INT type

9.27.3.12 `typedef ListElement* LinkList`

List of linkslinked list

Definition at line 1080 of file fasp.h.

9.27.3.13 `typedef struct linked_list ListElement`

Linked element in list

9.27.3.14 `typedef const grid2d* pcgrid2d`

Grid in 2d

Definition at line 1034 of file fasp.h.

9.27.3.15 `typedef grid2d* pgrid2d`

Grid in 2d

Definition at line 1032 of file fasp.h.

9.27.4 Variable Documentation

9.27.4.1 `INT* IMAP`

Red Black Gs Smoother imap

9.27.4.2 INT MAXIMAP

Red Black Gs Smoother max dofs of reservoir

9.27.4.3 INT nx_rb

Red Black Gs Smoother Nx

9.27.4.4 INT ny_rb

Red Black Gs Smoother Ny

9.27.4.5 INT nz_rb

Red Black Gs Smoother Nz

9.27.4.6 unsigned INT total_alloc_count

total allocation times

Definition at line 33 of file memory.c.

9.27.4.7 unsigned INT total_alloc_mem

total allocated memory

Definition at line 32 of file memory.c.

9.28 fasp_block.h File Reference

Main header file for FASP (block matrices)

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

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 diagonal preconditioning.*
- struct [precond_FASP_blkoi_data](#)
 - *Data passed to the preconditioner for preconditioning reservoir simulation problems.*

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_ivecator](#) [block_ivecator](#)
- typedef struct [block_Reservoir](#) [block_Reservoir](#)
- typedef struct [block_BSR](#) [block_BSR](#)
- typedef struct [precond_block_reservoir_data](#) [precond_block_reservoir_data](#)

9.28.1 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.

Definition in file [fasp_block.h](#).

9.28.2 Typedef Documentation

9.28.2.1 typedef struct [block_BSR](#) [block_BSR](#)

Block of BSR matrices of REAL type

9.28.2.2 typedef struct block_dCSRmat block_dCSRmat

Matrix of REAL type in Block CSR format

9.28.2.3 typedef struct block_dvector block_dvector

Vector of REAL type in Block format

9.28.2.4 typedef struct block_iCSRmat block_iCSRmat

Matrix of INT type in Block CSR format

9.28.2.5 typedef struct block_ivecator block_ivecator

Vector of INT type in Block format

9.28.2.6 typedef struct block_Reservoir block_Reservoir

Special block matrix for Reservoir Simulation

9.28.2.7 typedef struct dBSRmat dBSRmat

Matrix of REAL type in BSR format

9.28.2.8 typedef struct precondition_block_reservoir_data precondition_block_reservoir_data

Precond data for Reservoir Simulation

9.29 fmgcycle.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"
```

Functions

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

9.29.1 Detailed Description

Abstract non-recursive full multigrid cycle.

Definition in file [fmgcycle.c](#).

9.29.2 Function Documentation

9.29.2.1 void fasp_solver_fmgcycle (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.

Definition at line 34 of file fmgcycle.c.

9.30 formats.c File Reference

Matrix format conversion routines.

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

Functions

- [SHORT fasp_format_dcoo_dcsr](#) (dCOOmat *A, dCSRmat *B)
Transform a REAL matrix from its IJ format to its CSR format.
- [SHORT fasp_format_dcsr_dcoo](#) (dCSRmat *A, dCOOmat *B)
Transform a REAL matrix from its CSR format to its IJ format.
- [SHORT fasp_format_dstr_dcsr](#) (dSTRmat *A, dCSRmat *B)
Transfer a 'dSTRmat' type matrix into a 'dCSRmat' type matrix.
- [dCSRmat fasp_format_bdcsr_dcsr](#) (block_dCSRmat *Ab)
Form the whole dCSRmat A using blocks given in Ab.
- [dCSRmat * fasp_format_dcsr_dcsr](#) (dCSRmat *A)
Convert a dCSRmat into a dCSRmat.

- `dCSRmat fasp_format_dbsr_dcsr (dBSRmat *B)`
Transfer a '*dBSRmat*' type matrix into a *dCSRmat*.
- `dBSRmat fasp_format_dcsr_dbsr (dCSRmat *B, const INT nb)`
Transfer a *dCSRmat* type matrix into a *dBSRmat*.
- `dBSRmat fasp_format_dstr_dbsr (dSTRmat *B)`
Transfer a '*dSTRmat*' type matrix to a '*dBSRmat*' type matrix.
- `dCOOmat * fasp_format_dbsr_dcoo (dBSRmat *B)`
Transfer a '*dBSRmat*' type matrix to a '*dCOOmat*' type matrix.

9.30.1 Detailed Description

Matrix format conversion routines.

Definition in file [formats.c](#).

9.30.2 Function Documentation

9.30.2.1 `dCSRmat fasp_format_bdcsr_dcsr (block_dCSRmat * Ab)`

Form the whole *dCSRmat* A using blocks given in Ab.

Parameters

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

Returns

dCSRmat matrix if succeed, NULL if fail

Author

Shiquan Zhang

Date

08/10/2010

Definition at line 293 of file [formats.c](#).

9.30.2.2 `dCOOmat * fasp_format_dbsr_dcoo (dBSRmat * B)`

Transfer a '*dBSRmat*' type matrix to a '*dCOOmat*' type matrix.

Parameters

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

Returns

Pointer to *dCOOmat* matrix

Author

Zhiyang Zhou

Date

2010/10/26

Definition at line 956 of file formats.c.

9.30.2.3 dCSRmat fasp_format_dbsr_dcsr (dBSRmat * B)

Transfer a 'dBSRmat' type matrix into a dCSRmat.

Parameters

<i>B</i>	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.

9.30.2.4 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

SUCCESS if succeed

Author

Xuehai Huang

Date

08/10/2009

Definition at line 28 of file formats.c.

9.30.2.5 dBSRmat fasp_format_dcsr_dbsr (dCSRmat * B, const INT nb)

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

Parameters

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

Returns

[dBSRmat](#) matrix

Author

Changhe Qiao

Date

03/12/2012

Modified by Xiaozhe Hu on 03/13/2012 Modified by Chunsheng Feng, Zheng Li on 10/13/2012

Definition at line 721 of file formats.c.

9.30.2.6 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

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.

9.30.2.7 dCSRLmat * fasp_format_dcsrl_dcsr (dCSRmat * A)

Convert a [dCSRmat](#) into a [dCSRLmat](#).

Parameters

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

Returns

Pointer to [dCSRLmat](#) matrix

Author

Zhiyang Zhou

Date

2011/01/07

Definition at line 362 of file formats.c.

9.30.2.8 [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 852 of file formats.c.

9.30.2.9 [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

SUCCESS if succeed

Author

Zhiyang Zhou

Date

2010/04/29

Definition at line 118 of file formats.c.

9.31 givens.c File Reference

Givens transformation.

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

Functions

- void [fasp_aux_givens](#) (const [REAL](#) *beta*, [dCSRmat](#) **H*, [dvector](#) **y*, [REAL](#) **tmp*)
Perform Givens rotations to compute $y \mid \beta e_1 - H*y$.

9.31.1 Detailed Description

Givens transformation.

Definition in file [givens.c](#).

9.31.2 Function Documentation

9.31.2.1 void [fasp_aux_givens](#) (const [REAL](#) *beta*, [dCSRmat](#) * *H*, [dvector](#) * *y*, [REAL](#) * *tmp*)

Perform Givens rotations to compute $y \mid \beta e_1 - H*y$.**Parameters**

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

Author

Xuehai Huang

Date

10/19/2008

Definition at line 28 of file givens.c.

9.32 gmg_poisson.c File Reference

GMG method as an iterative solver for Poisson Problem.

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

Functions

- **INT fasp_poisson_gmg_1D** (**REAL** *u, **REAL** *b, **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_fgmg_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_fgmg_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_fgmg_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)

9.32.1 Detailed Description

GMG method as an iterative solver for Poisson Problem.

Definition in file [gmg_poisson.c](#).

9.32.2 Function Documentation

9.32.2.1 void fasp_poisson_fgmg_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

<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 417 of file gmg_poisson.c.

9.32.2.2 void fasp_poisson_fgmg_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)

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 509 of file gmg_poisson.c.

9.32.2.3 void fasp_poisson_fgmg_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 616 of file gmg_poisson.c.

9.32.2.4 **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.

9.32.2.5 **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 154 of file gmg_poisson.c.

9.32.2.6 **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 285 of file gmg_poisson.c.

9.32.2.7 **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 723 of file gmg_poisson.c.

9.32.2.8 `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 813 of file gmg_poisson.c.

9.32.2.9 `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 918 of file gmg_poisson.c.

9.33 gmg_util.inl File Reference

Routines for GMG solvers.

9.33.1 Detailed Description

Routines for GMG solvers.

Definition in file [gmg_util.inl](#).

9.34 graphics.c File Reference

Functions for graphical output.

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

Functions

- void [fasp_dcsr_subplot](#) (const [dCSRmat](#) *A, const char *filename, [INT](#) size)
Write sparse matrix pattern in BMP file format.
- void [fasp_dbsr_subplot](#) (const [dBSRmat](#) *A, const char *filename, [INT](#) size)
- 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)

9.34.1 Detailed Description

Functions for graphical output.

Definition in file [graphics.c](#).

9.34.2 Function Documentation

9.34.2.1 void fasp_dbsr_plot (const dBSRmat * *A*, const char * *filename*)

Write dBSR sparse matrix pattern in BMP file format.

Parameters

<i>A</i>	Pointer to the dBSRmat 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 466 of file graphics.c.

9.34.2.2 void fasp_dcsr_subplot (const dCSRmat * *A*, const char * *filename*, INT *size*)

Write sparse matrix pattern in BMP file format.

Parameters

<i>A</i>	Pointer to the dCSRmat matrix
<i>filename</i>	File name
<i>size</i>	size*size is the picture size for the picture

Author

Chensong Zhang

Date

03/29/2009

Note

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

Each pixel corresponds to one matrix element. The pixel colors have the following meaning:

White structurally zero element Blue positive element Red negative element Brown nearly zero element

Definition at line 44 of file `graphics.c`.

9.34.2.3 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 175 of file `graphics.c`.

9.35 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 **srtr** (num, q)
- subroutine **qsplrit** (a, ind, n, ncut)
- subroutine **symbfactor** (n, colind, rwptr, levfill, nzmax, nzlu, ijlu, uptr, ierr)

9.35.1 Detailed Description

ILU routines for preconditioning adapted from SPARSEKIT.

Note

Incomplete Factorization Methods: ILUk, ILUt, ILUtp

Definition in file [ilu.f](#).

9.36 ilu_setup_bsr.c File Reference

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

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

Functions

- void **symbfactor_** (const [INT](#) *n, [INT](#) *colind, [INT](#) *rowptr, 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.

9.36.1 Detailed Description

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

Definition in file [ilu_setup_bsr.c](#).

9.36.2 Function Documentation

9.36.2.1 [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](#).

9.37 `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"
```

Functions

- void `iluk_` (const `INT` *n, `REAL` *a, `INT` *ja, `INT` *ia, `INT` *lfil, `REAL` *alu, `INT` *jlu, `INT` *iwk, `INT` *ierr, `INT` *nzlu)
- void `ilut_` (const `INT` *n, `REAL` *a, `INT` *ja, `INT` *ia, `INT` *lfil, const `REAL` *droptol, `REAL` *alu, `INT` *jlu, `INT` *iwk, `INT` *ierr, `INT` *nz)
- void `ilutp_` (const `INT` *n, `REAL` *a, `INT` *ja, `INT` *ia, `INT` *lfil, const `REAL` *droptol, const `REAL` *permtol, const `INT` *mbloc, `REAL` *alu, `INT` *jlu, `INT` *iwk, `INT` *ierr, `INT` *nz)
- `SHORT fasp_ilu_dcsr_setup` (`dCSRmat` *A, `ILU_data` *iludata, `ILU_param` *iluparam)

Get ILU decoposition of a CSR matrix A.

9.37.1 Detailed Description

Setup of ILU decomposition for `dCSRmat` matrices.

Definition in file `ilu_setup_csr.c`.

9.37.2 Function Documentation

9.37.2.1 `SHORT fasp_ilu_dcsr_setup` (`dCSRmat` * A, `ILU_data` * *iludata*, `ILU_param` * *iluparam*)

Get ILU decoposition of a CSR matrix A.

Parameters

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

Author

Shiquan Zhang

Date

12/27/2009

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

9.38 `ilu_setup_str.c` File Reference

Setup of ILU decomposition for `dSTRmat` matrices.

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

Functions

- void [fasp_ilu_dstr_setup0](#) ([dSTRmat](#) *A, [dSTRmat](#) *LU)
Get ILU(0) decomposition of a structured matrix A.
- void [fasp_ilu_dstr_setup1](#) ([dSTRmat](#) *A, [dSTRmat](#) *LU)
Get ILU(1) decomposition of a structured matrix A.

9.38.1 Detailed Description

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

Definition in file [ilu_setup_str.c](#).

9.38.2 Function Documentation

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

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

Parameters

<i>A</i>	Pointer to dSTRmat
<i>LU</i>	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](#).

9.38.2.2 void [fasp_ilu_dstr_setup1](#) ([dSTRmat](#) * A, [dSTRmat](#) * LU)

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

Parameters

<i>A</i>	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`.

9.39 init.c File Reference

Initialize important data structures.

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

Functions

- void `fasp_precond_data_null` (`precond_data` *pcdata)
Initialize `precond_data`.
- `AMG_data` * `fasp_amg_data_create` (`SHORT` max_levels)
Create and initialize `AMG_data` for classical and SA AMG.
- `AMG_data_bsr` * `fasp_amg_data_bsr_create` (`SHORT` max_levels)
Create and initialize `AMG_data` data structure 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 memory space.
- void `fasp_amg_data_free` (`AMG_data` *mgl, `AMG_param` *param)
Free `AMG_data` data memory space.
- void `fasp_amg_data_bsr_free` (`AMG_data_bsr` *mgl)
Free `AMG_data_bsr` data memory space.
- void `fasp_ilu_data_free` (`ILU_data` *ILUdata)
Create `ILU_data` structure.
- void `fasp_ilu_data_null` (`ILU_data` *ILUdata)
Initialize ILU data.
- void `fasp_precond_null` (`precond` *pcdata)
Initialize precondition data.

9.39.1 Detailed Description

Initialize important data structures.

Note

Every structures should be initialized before usage.

Definition in file [init.c](#).

9.39.2 Function Documentation

9.39.2.1 `AMG_data_bsr * fasp_amg_data_bsr_create (SHORT max_levels)`

Create and initialize [AMG_data](#) data sturcture 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 83 of file `init.c`.

9.39.2.2 `void fasp_amg_data_bsr_free (AMG_data_bsr * mgl)`

Free [AMG_data_bsr](#) data memeory space.

Parameters

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

Author

Xiaozhe Hu

Date

2013/02/13

Definition at line 213 of file `init.c`.

9.39.2.3 `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
-------------------	------------------------------

Returns

Pointer to the [AMG_data](#) data structure

Author

Chensong Zhang

Date

2010/04/06

Definition at line 56 of file init.c.

9.39.2.4 void fasp_amg_data_free ([AMG_data](#) * *mgl*, [AMG_param](#) * *param*)

Free [AMG_data](#) data memeory 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 166 of file init.c.

9.39.2.5 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 111 of file init.c.

9.39.2.6 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 258 of file init.c.

9.39.2.7 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 279 of file init.c.

9.39.2.8 void fasp_precond_data_null (precondition_data * pcddata)

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.

9.39.2.9 void fasp_precond_null (precondition * pcddata)

Initialize precondition data.

Parameters

<i>pcdata</i>	Pointer to precondition
---------------	-------------------------

Author

Chensong Zhang

Date

2010/03/23

Definition at line 295 of file init.c.

9.39.2.10 void fasp_schwarz_data_free (Schwarz_data * schwarz)

Free Schwarz_data data memory space.

Parameters

*schwarz	pointer to the AMG_data data
----------	------------------------------

Author

Xiaozhe Hu

Date

2010/04/06

Definition at line 137 of file init.c.

9.40 input.c File Reference

Read input parameters.

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

Functions

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

9.40.1 Detailed Description

Read input parameters.

Definition in file [input.c](#).

9.40.2 Function Documentation

9.40.2.1 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.

9.40.2.2 void fasp_param_input (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 95 of file input.c.

9.41 interface_mumps.c File Reference

Interface to MUMPS direct solvers.

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

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, const int job)
Solve $Ax=b$ by MUMPS in three steps.

9.41.1 Detailed Description

Interface to MUMPS direct solvers.

Definition in file [interface_mumps.c](#).

9.41.2 Function Documentation

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

Solve $Ax=b$ by MUMPS directly.

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>print_level</i>	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](#).

9.41.2.2 int [fasp_solver_mumps_steps](#) (dCSRmat * *ptrA*, dvector * *b*, dvector * *u*, const int *job*)

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>job</i>	1: Setup, 2: Solve, 3 Destroy

Author

Chunsheng Feng

Date

02/27/2013

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

Definition at line 161 of file interface_mumps.c.

9.42 interface_samg.c File Reference

Interface to SAMG.

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

Functions

- void [dvector2SAMGInput](#) ([dvector](#) *vec, char *filename)
Write a dvector to disk file in SAMG format (coordinate format)
- [INT dCSRmat2SAMGInput](#) ([dCSRmat](#) *A, char *filefrm, char *fileamg)
Write SAMG Input data from a sparse matrix of CSR format.

9.42.1 Detailed Description

Interface to SAMG. Add reference for SAMG by K. Stuben here!

Definition in file [interface_samg.c](#).

9.42.2 Function Documentation

9.42.2.1 INT dCSRmat2SAMGInput (dCSRmat * A, char * filefrm, char * fileamg)

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

Parameters

*A	pointer to the dCSRmat matrix
*filefrm	pointer to the name of the .frm file
*fileamg	pointer to the name of the .amg file

Author

Zhiyang Zhou

Date

2010/08/25

Definition at line 56 of file interface_samg.c.

9.42.2.2 void dvector2SAMGInput (dvector * *vec*, char * *filename*)

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

Parameters

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

Author

Zhiyang Zhou

Date

08/25/2010

Definition at line 27 of file interface_samg.c.

9.43 interface_superlu.c File Reference

Interface to SuperLU direct solvers.

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

Functions

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

9.43.1 Detailed Description

Interface to SuperLU direct solvers.

Definition in file [interface_superlu.c](#).

9.43.2 Function Documentation

9.43.2.1 int [fasp_solver_superlu](#) (dCSRmat * *ptrA*, dvector * *b*, dvector * *u*, const int *print_level*)

Solve $Au=b$ by SuperLU.

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>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.

9.44 interface_umfpack.c File Reference

Interface to UMFPACK direct solvers.

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

Functions

- int [fasp_solver_umfpack](#) ([dCSRmat](#) *ptrA, [dvector](#) *b, [dvector](#) *u, const int print_level)
Solve $Au=b$ by UMFPack.

9.44.1 Detailed Description

Interface to UMFPACK direct solvers.

Definition in file [interface_umfpack.c](#).

9.44.2 Function Documentation

9.44.2.1 int [fasp_solver_umfpack](#) ([dCSRmat](#) * *ptrA*, [dvector](#) * *b*, [dvector](#) * *u*, const int *print_level*)

Solve $Au=b$ by UMFPack.

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>print_level</i>	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.

9.45 interpolation.c File Reference

Interpolation operators for AMG.

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

Functions

- void [fasp_amg_interp](#) ([dCSRmat](#) *A, [ivector](#) *vertices, [dCSRmat](#) *P, [iCSRmat](#) *S, [AMG_param](#) *param)
Generate interpolation operator P.
- void [fasp_amg_interp1](#) ([dCSRmat](#) *A, [ivector](#) *vertices, [dCSRmat](#) *P, [AMG_param](#) *param, [iCSRmat](#) *S, [INT](#) *icor_ysk)
Generate interpolation operator P.
- void [fasp_amg_interp_trunc](#) ([dCSRmat](#) *P, [AMG_param](#) *param)
Truncation step for prolongation operators.

9.45.1 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.

Definition in file [interpolation.c](#).

9.45.2 Function Documentation

9.45.2.1 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.

9.45.2.2 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

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.

9.45.2.3 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 158 of file interpolation.c.

9.46 interpolation_em.c File Reference

Interpolation operators for AMG based on energy-min.

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

Functions

- void [fasp_amg_interp_em](#) ([dCSRmat](#) *A, [ivector](#) *vertices, [dCSRmat](#) *P, [AMG_param](#) *param)
Energy-min interpolation.

9.46.1 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

Definition in file [interpolation_em.c](#).

9.46.2 Function Documentation

9.46.2.1 void [fasp_amg_interp_em](#) ([dCSRmat](#) * A, [ivector](#) * vertices, [dCSRmat](#) * P, [AMG_param](#) * param)

Energy-min interpolation.

Parameters

<i>A</i>	Pointer to dCSRmat : 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 dCSRmat matrix of resulted interpolation
<i>param</i>	Pointer to AMG_param : AMG parameters

Author

Shuo Zhang, Xuehai Huang

Date

04/04/2010

Modified by Chunsheng Feng, Zheng Li on 10/17/2012: add OMP support Modified by Chensong Zhang on 05/14/2013: reconstruct the code

Definition at line 49 of file interpolation_em.c.

9.47 io.c File Reference

Matrix-vector input/output subroutines.

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

Functions

- void [fasp_dcsrvec1_read](#) (char *filename, [dCSRmat](#) *A, [dvector](#) *b)
Read A and b from a SINGLE disk file.
- void [fasp_dcsrvec2_read](#) (char *filemat, char *filerhs, [dCSRmat](#) *A, [dvector](#) *b)
Read A and b from two disk files.
- void [fasp_dcsr_read](#) (char *filename, [dCSRmat](#) *A)
Read A from matrix disk file in IJ format.
- void [fasp_dcoo_read](#) (char *filename, [dCSRmat](#) *A)
Read A from matrix disk file in IJ format – indices starting from 0.
- void [fasp_dcoo1_read](#) (char *filename, [dCOOmat](#) *A)
Read A from matrix disk file in IJ format – indices starting from 0.
- void [fasp_dmtx_read](#) (char *filename, [dCSRmat](#) *A)
Read A from matrix disk file in MatrixMarket general format.
- void [fasp_dmtxsym_read](#) (char *filename, [dCSRmat](#) *A)
Read A from matrix disk file in MatrixMarket sym format.
- void [fasp_dstr_read](#) (char *filename, [dSTRmat](#) *A)
Read A from a disk file in dSTRmat format.
- void [fasp_dbsr_read](#) (char *filename, [dBSRmat](#) *A)
Read A from a disk file in dBSRmat format.
- void [fasp_dvecind_read](#) (char *filename, [dvector](#) *b)

- Read b from matrix disk file.*

 - void `fasp_dvec_read` (char *filename, `dvector` *b)

Read b from a disk file in array format.
- void `fasp_ivecind_read` (char *filename, `ivector` *b)

Read b from matrix disk file.
- void `fasp_ivec_read` (char *filename, `ivector` *b)

Read b from a disk file in array format.
- void `fasp_dcsrvec1_write` (char *filename, `dCSRmat` *A, `dvector` *b)

Write A and b to a SINGLE disk file.
- void `fasp_dcsrvec2_write` (char *filename, char *filerhs, `dCSRmat` *A, `dvector` *b)

Write A and b to two disk files.
- void `fasp_dcoo_write` (char *filename, `dCSRmat` *A)

Write a matrix to disk file in IJ format (coordinate format)
- void `fasp_dstr_write` (char *filename, `dSTRmat` *A)

Write a dSTRmat to a disk file.
- void `fasp_dbsr_write` (char *filename, `dBsrmat` *A)

Write a dBsrmat to a disk file.
- void `fasp_dvec_write` (char *filename, `dvector` *vec)

Write a dvector to disk file.
- void `fasp_dvecind_write` (char *filename, `dvector` *vec)

Write a dvector to disk file in coordinate format.
- void `fasp_ivec_write` (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)
- void `fasp_dstr_print` (`dSTRmat` *A)

Print out a dSTRmat matrix in coordinate format.
- void `fasp_matrix_read` (char *filename, void *A)

Read matrix from different kinds of formats from both ASCII and binary files.
- void `fasp_matrix_read_bin` (char *filename, void *A)

Read matrix in binary format.
- void `fasp_matrix_write` (char *filename, void *A, INT flag)

write matrix from different kinds of formats from both ASCII and binary files
- void `fasp_vector_read` (char *filerhs, void *b)

Read RHS vector from different kinds of formats from both ASCII and binary files.
- void `fasp_vector_write` (char *filerhs, void *b, INT flag)

write RHS vector from different kinds of formats in both ASCII and binary files

Variables

- [INT ilength](#)
Flags which indicates lengths of INT and REAL numbers.
- [INT dlength](#)

9.47.1 Detailed Description

Matrix-vector input/output subroutines.

Note

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

Definition in file [io.c](#).

9.47.2 Function Documentation

9.47.2.1 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 1384 of file io.c.

9.47.2.2 void fasp_dbsr_read (char * filename, dBSRmat * A)

Read A from a disk file in [dBSRmat](#) format.

Parameters

<i>filename</i>	File name for matrix A
<i>A</i>	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 635 of file io.c.

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

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

Parameters

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

Note

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

Author

Shiquan Zhang

Date

10/29/2010

Definition at line 1143 of file io.c.

9.47.2.4 void fasp_dbsr_write_coo (const char * *filename*, const [dBSRmat](#) * *A*)

Print out a [dBSRmat](#) matrix in coordinate format for matlab spy.

Parameters

<i>A</i>	Pointer to the dBSRmat matrix <i>A</i>
----------	--

Author

Chunsheng Feng

Date

11/14/2013

Definition at line 1418 of file io.c.

9.47.2.5 void fasp_dcoo1_read (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
<i>A</i>	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 function is this function do not change to CSR format

Author

Xiaozhe Hu

Date

03/24/2013

Definition at line 366 of file io.c.

9.47.2.6 void fasp_dcoo_print (dCOOmat * *A*)

Print out a [dCOOmat](#) matrix in coordinate format.

Parameters

<i>A</i>	Pointer to the dCOOmat matrix <i>A</i>
----------	--

Author

Ziteng Wang

Date

12/24/2012

Definition at line 1362 of file io.c.

9.47.2.7 void fasp_dcoo_read (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 315 of file io.c.

9.47.2.8 void fasp_dcoo_write (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 1043 of file io.c.

9.47.2.9 void fasp_dcsr_print (dCSRmat * A)

Print out a [dCSRmat](#) matrix in coordinate format.

Parameters

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

Author

Xuehai Huang

Date

03/29/2009

Definition at line 1340 of file io.c.

9.47.2.10 void fasp_dcsr_read (char * *filename*, [dCSRmat](#) * *A*)

Read *A* from matrix disk file in IJ format.

Parameters

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

Author

Ziteng Wang

Date

12/25/2012

Definition at line 254 of file io.c.

9.47.2.11 void fasp_dcsrvec1_read (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 84 of file io.c.

9.47.2.12 void fasp_dcsrvec1_write (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 Zhang on 12/26/2012

Definition at line 895 of file io.c.

9.47.2.13 void fasp_dcsrvec2_read (char * *filemat*, 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 175 of file io.c.

9.47.2.14 void fasp_dcsrvec2_write (char * *filemat*, 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 972 of file io.c.

9.47.2.15 void fasp_dmtx_read (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 dCS-Rmat format. For details of mtx format, please refer to <http://math.nist.gov/MatrixMarket/>.
Indices start from 1, NOT 0!!!

Author

Chensong Zhang

Date

09/05/2011

Definition at line 416 of file io.c.

9.47.2.16 void fasp_dmtxsym_read (char * *filename*, dCSRmat * A)

Read A from matrix disk file in MatrixMarket sym format.

Parameters

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

<i>A</i>	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 478 of file io.c.

9.47.2.17 void fasp_dstr_print (dSTRmat * A)

Print out a [dSTRmat](#) matrix in coordinate format.

Parameters

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

Author

Ziteng Wang

Date

12/24/2012

Definition at line 1501 of file io.c.

9.47.2.18 void fasp_dstr_read (char * filename, dSTRmat * A)

Read A from a disk file in [dSTRmat](#) format.

Parameters

<i>filename</i>	File name for the matrix
<i>A</i>	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
- $\text{diag}(j)$, $j=0:n-1$
- offset, length: offset and length of off-diag1
- $\text{offdiag}(j)$, $j=0:\text{length}-1$

Author

Xuehai Huang

Date

03/29/2009

Definition at line 555 of file io.c.

9.47.2.19 `void fasp_dstr_write (char * filename, dSTRmat * A)`

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

Parameters

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

Note

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

Author

Shiquan Zhang

Date

03/29/2010

Definition at line 1083 of file io.c.

9.47.2.20 `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 1301 of file io.c.

9.47.2.21 void fasp_dvec_read (char * *filename*, dvector * *b*)

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

Parameters

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

Note

File Format:

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

Author

Chensong Zhang

Date

03/29/2009

Definition at line 754 of file io.c.

9.47.2.22 void fasp_dvec_write (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 1198 of file io.c.

9.47.2.23 void fasp_dvecind_read (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 704 of file io.c.

9.47.2.24 void fasp_dvecind_write (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 1234 of file io.c.

9.47.2.25 void fasp_ivec_print (INT *n*, ivec * *u*)

Print first n entries of a vector of INT type.

Parameters

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

Author

Chensong Zhang

Date

03/29/2009

Definition at line 1321 of file io.c.

9.47.2.26 void fasp_ivec_read (char * *filename*, ivector * *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 844 of file io.c.

9.47.2.27 void fasp_ivec_write (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 1269 of file io.c.

9.47.2.28 void fasp_ivecind_read (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 804 of file io.c.

9.47.2.29 fasp_matrix_read (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 1535 of file io.c.

9.47.2.30 void fasp_matrix_read_bin (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 1640 of file io.c.

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

write 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
<i>flag</i>	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 1714 of file io.c.

9.47.2.32 fasp_vector_read (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
- dlength % third digit of formatflag, length of REAL

Author

Ziteng Wang

Date

12/24/2012

Definition at line 1807 of file io.c.

9.47.2.33 fasp_vector_write (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 1904 of file io.c.

9.48 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"
```

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.

9.48.1 Detailed Description

Iterative solvers for [block_dCSRmat](#) matrices.

Definition in file [itsolver_bcsr.c](#).

9.48.2 Function Documentation

9.48.2.1 [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](#).

9.48.2.2 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 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

Returns

Number of iterations if succeed

Author

Xiaozhe Hu

Date

07/18/2010

Definition at line 105 of file itsolver_bcsr.c.

9.49 itsolver_bsr.c File Reference

Iterative solvers for [dBSRmat](#) matrices.

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

Functions

- 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++.
- 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.

Variables

- [INT THDs_AMG_GS](#) =0
- [INT THDs_CPR_IGS](#) =0
- [INT THDs_CPR_gGS](#) =0

9.49.1 Detailed Description

Iterative solvers for [dBSRmat](#) matrices.

Definition in file [itsolver_bsr.c](#).

9.49.2 Function Documentation

9.49.2.1 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 sovler
<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 39 of file [itsolver_bsr.c](#).

9.49.2.2 INT fasp_solver_dbsr_itsolver (dBSRmat * *A*, dvector * *b*, dvector * *x*, precondition * *pc*, itsolver_param * *itparam*)

Solve $Ax=b$ by preconditioned Krylov methods for BSR matrices.

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>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 100 of file itsolver_bsr.c.

9.49.2.3 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 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

Returns

Number of iterations if succeed

Author

Zhiyang Zhou, Xiaozhe Hu

Date

10/26/2010

Definition at line 175 of file itsolver_bsr.c.

9.49.2.4 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 382 of file itsolver_bsr.c.

9.49.2.5 INT fasp_solver_dbsr_krylov_diag (dBSRmat * *A*, dvector * *b*, dvector * *x*, itsolver_param * *itparam*)Solve $Ax=b$ by diagonal preconditioned Krylov methods.

Parameters

<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

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 219 of file itsolver_bsr.c.

9.49.2.6 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 315 of file itsolver_bsr.c.

9.49.3 Variable Documentation**9.49.3.1 INT THDs_AMG_GS =0**

cpr amg gs smoothing threads

Definition at line 35 of file itsolver_bsr.c.

9.49.3.2 INT THDs_CPR_gGS =0

global matrix gs smoothing threads

Definition at line 37 of file itsolver_bsr.c.

9.49.3.3 INT THDs_CPR_IGS =0

reservoir gs smoothing threads

Definition at line 36 of file itsolver_bsr.c.

9.50 itsolver_csr.c File Reference

Iterative solvers for [dCSRmat](#) matrices.

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

Functions

- [INT fasp_solver_dcsr_itsolver](#) ([dCSRmat](#) *A, [dvector](#) *b, [dvector](#) *x, [precond](#) *pc, [itsolver_param](#) *itparam)
Solve $Ax=b$ by preconditioned Krylov methods for CSR matrices.
- [INT fasp_solver_dcsr_krylov](#) ([dCSRmat](#) *A, [dvector](#) *b, [dvector](#) *x, [itsolver_param](#) *itparam)
Solve $Ax=b$ by standard Krylov methods for CSR matrices.
- [INT fasp_solver_dcsr_krylov_diag](#) ([dCSRmat](#) *A, [dvector](#) *b, [dvector](#) *x, [itsolver_param](#) *itparam)
Solve $Ax=b$ by diagonal preconditioned Krylov methods.
- [INT fasp_solver_dcsr_krylov_schwarz](#) ([dCSRmat](#) *A, [dvector](#) *b, [dvector](#) *x, [itsolver_param](#) *itparam, [Schwarz_param](#) *schparam)
Solve $Ax=b$ by overlapping schwarz Krylov methods.
- [INT fasp_solver_dcsr_krylov_amg](#) ([dCSRmat](#) *A, [dvector](#) *b, [dvector](#) *x, [itsolver_param](#) *itparam, [AMG_param](#) *amgparam)
Solve $Ax=b$ by AMG preconditioned Krylov methods.
- [INT fasp_solver_dcsr_krylov_ilu](#) ([dCSRmat](#) *A, [dvector](#) *b, [dvector](#) *x, [itsolver_param](#) *itparam, [ILU_param](#) *iluparam)
Solve $Ax=b$ by ILUs preconditioned Krylov methods.
- [INT fasp_solver_dcsr_krylov_ilu_M](#) ([dCSRmat](#) *A, [dvector](#) *b, [dvector](#) *x, [itsolver_param](#) *itparam, [ILU_param](#) *iluparam, [dCSRmat](#) *M)
Solve $Ax=b$ by ILUs preconditioned Krylov methods: ILU of M as preconditioner.

9.50.1 Detailed Description

Iterative solvers for [dCSRmat](#) matrices.

Definition in file [itsolver_csr.c](#).

9.50.2 Function Documentation

9.50.2.1 [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.

9.50.2.2 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

<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 139 of file itsolver_csr.c.

9.50.2.3 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 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 methods

Returns

Number of iterations if succeed

Author

Chensong Zhang

Date

09/25/2009

Definition at line 330 of file itsolver_csr.c.

9.50.2.4 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 189 of file itsolver_csr.c.

9.50.2.5 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 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>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 438 of file itsolver_csr.c.

9.50.2.6 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 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>iluparam</i>	Pointer to parameters for ILU
<i>M</i>	Pointer to the preconditioning matrix in dCSRmat 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 525 of file itsolver_csr.c.

9.50.2.7 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

<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
<i>schparam</i>	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 253 of file itsolver_csr.c.

9.51 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"
```

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.

9.51.1 Detailed Description

Iterative solvers with matrix-free spmv.

Definition in file [itsolver_mf.c](#).

9.51.2 Function Documentation

9.51.2.1 **INT** fasp_solver_itsolver (mxv_matfree * *mf*, dvector * *b*, dvector * *x*, precondition * *pc*, itsolver_param * *itparam*)

Solve $Ax=b$ by preconditioned Krylov methods for CSR matrices.

Parameters

<i>mf</i>	Pointer to mxv_matfree 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>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.

9.51.2.2 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 mxv_matfree 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.

9.51.2.3 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 mxv_matfree 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.

9.52 [itsolver_str.c](#) File Reference

Iterative solvers for [dSTRmat](#) matrices.

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

Functions

- [INT fasp_solver_dstr_itsolver](#) ([dSTRmat](#) *A, [dvector](#) *b, [dvector](#) *x, [precond](#) *pc, [itsolver_param](#) *itparam)
Solve $Ax=b$ by standard Krylov methods.
- [INT fasp_solver_dstr_krylov](#) ([dSTRmat](#) *A, [dvector](#) *b, [dvector](#) *x, [itsolver_param](#) *itparam)
Solve $Ax=b$ by standard Krylov methods.
- [INT fasp_solver_dstr_krylov_diag](#) ([dSTRmat](#) *A, [dvector](#) *b, [dvector](#) *x, [itsolver_param](#) *itparam)
Solve $Ax=b$ by diagonal preconditioned Krylov methods.
- [INT fasp_solver_dstr_krylov_ilu](#) ([dSTRmat](#) *A, [dvector](#) *b, [dvector](#) *x, [itsolver_param](#) *itparam, [ILU_param](#) *iluparam)
Solve $Ax=b$ by structured ILU preconditioned Krylov methods.
- [INT fasp_solver_dstr_krylov_blockgs](#) ([dSTRmat](#) *A, [dvector](#) *b, [dvector](#) *x, [itsolver_param](#) *itparam, [ivector](#) *neigh, [ivector](#) *order)
Solve $Ax=b$ by diagonal preconditioned Krylov methods.

9.52.1 Detailed Description

Iterative solvers for [dSTRmat](#) matrices.

Definition in file [itsolver_str.c](#).

9.52.2 Function Documentation

9.52.2.1 `INT fasp_solver_dstr_itsolver (dSTRmat * A, dvector * b, dvector * x, precondition * pc, 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>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](#).

9.52.2.2 `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 109 of file itsolver_str.c.

9.52.2.3 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 290 of file itsolver_str.c.

9.52.2.4 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 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

4/23/2010

Definition at line 149 of file itsolver_str.c.

9.52.2.5 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 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>iluparam</i>	Pointer to parameters for ILU

Returns

Number of iterations if succeed

Author

Xiaozhe Hu

Date

05/01/2010

Definition at line 207 of file itsolver_str.c.

9.53 itsolver_util.inl File Reference

Routines for iterative solvers.

Macros

- #define **ITS_FACONV** printf("### WARNING: False convergence!\n")
Warning for residual false convergence.
- #define **ITS_ZEROSOL** printf("### WARNING: Iteration stopped due to the solution is close to zero!\n")
Warning for solution close to zero.
- #define **ITS_RESTART** printf("### WARNING: Iteration restarted due to stagnation!\n")
Warning for iteration restarted.
- #define **ITS_STAGGED** printf("### WARNING: Iteration stopped due to staggnation!\n")
Warning for staged iteration.
- #define **ITS_ZEROTOL** printf("### WARNING: The tolerance might be too small!\n")
Warning for tolerance practically close to zero.
- #define **ITS_DIVZERO** printf("### WARNING: Divided by zero!\n")
Warning for divided by zero.
- #define **ITS_REALRES**(relres) printf("### WARNING: The actual relative residual = %e!\n",(relres))
Warning for actual relative residual.
- #define **ITS_COMPRES**(relres) printf("### WARNING: The computed relative residual = %e!\n",(relres))
Warning for computed relative residual.

- `#define ITS_SMALLSP` `printf("### WARNING: The sp is too small!\n")`
Warning for too small sp.
- `#define ITS_RESTORE`(iter) `printf("### WARNING: Restore iteration %d!!!", (iter));`
Warning for restore previous iteration.
- `#define ITS_DIFFRES`(reldiff, relres) `printf("||u-u'|| = %e and the comp. rel. res. = %e.\n", (reldiff), (relres));`
Output relative difference and residual.
- `#define ITS_PUTNORM`(name, value) `printf("L2 norm of %s = %e.\n", (name), (value));`
Output L2 norm of some variable.

9.53.1 Detailed Description

Routines for iterative solvers.

Definition in file [itsolver_util.inl](#).

9.54 linklist.inl File Reference

Utilities for link list data structure.

Macros

- `#define LIST_HEAD` -1
- `#define LIST_TAIL` -2

9.54.1 Detailed Description

Utilities for link list data structure.

Note

These linked-list operations are adapted from hypre 2.0

Definition in file [linklist.inl](#).

9.54.2 Macro Definition Documentation

9.54.2.1 `#define LIST_HEAD` -1

head of the linked list

Definition at line 7 of file [linklist.inl](#).

9.54.2.2 `#define LIST_TAIL` -2

tail of the linked list

Definition at line 8 of file [linklist.inl](#).

9.55 lu.c File Reference

LU decomposition and direct solve for dense matrix.

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

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.

9.55.1 Detailed Description

LU decomposition and direct solve for dense matrix.

Definition in file [lu.c](#).

9.55.2 Function Documentation

9.55.2.1 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

SUCCESS if succeed, RUN_FAIL 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.

9.55.2.2 `SHORT fasp_smat_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

SUCCESS if succeed, RUN_FAIL 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.

9.56 `memory.c` File Reference

Memory allocation and deallocation.

`#include "fasp.h"`

Functions

- void * **fasp_mem_calloc** (LONGLONG size, INT type)
- void * **fasp_mem_realloc** (void *oldmem, LONG 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, char *message, **INT** ERR)
Check wether a point is null or not.
- **SHORT** [fasp_mem_iludata_check](#) (**ILU_data** *iludata)
Check wether a [ILU_data](#) has enough work space.
- **SHORT** [fasp_mem_dcsr_check](#) (**dCSRmat** *A)
Check wether a [dCSRmat](#) A has sucessfully allocated memory.

Variables

- unsigned **INT** [total_alloc_mem](#) = 0
- unsigned **INT** [total_alloc_count](#) = 0

9.56.1 Detailed Description

Memory allocation and deallocation.

Definition in file [memory.c](#).

9.56.2 Function Documentation

9.56.2.1 **SHORT** [fasp_mem_check](#) (void * *ptr*, char * *message*, **INT** *ERR*)

Check wether 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

SUCCESS or error code

Author

Chensong Zhang

Date

11/16/2009

Definition at line 191 of file [memory.c](#).

9.56.2.2 **SHORT** [fasp_mem_dcsr_check](#) (**dCSRmat** * *A*)

Check wether a [dCSRmat](#) A has sucessfully allocated memory.

Parameters

<i>A</i>	Pointer to be cheked
----------	----------------------

Returns

SUCCESS if success, else ERROR message (negative value)

Author

Xiaozhe Hu

Date

11/27/09

Definition at line 241 of file memory.c.

9.56.2.3 void fasp_mem_free (void * *mem*)

Free up previous allocated memory body.

Parameters

<i>mem</i>	Pointer to the memory body need to be freed
------------	---

Returns

NULL pointer

Author

Chensong Zhang

Date

2010/12/24

Definition at line 144 of file memory.c.

9.56.2.4 SHORT fasp_mem_iludata_check (ILU_data * *iludata*)

Check wether a [ILU_data](#) has enough work space.

Parameters

<i>iludata</i>	Pointer to be cheked
----------------	----------------------

Returns

SUCCESS if success, else ERROR (negative value)

Author

Xiaozhe Hu, Chensong Zhang

Date

11/27/09

Definition at line 215 of file memory.c.

9.56.2.5 void * fasp_mem_realloc (void * *oldmem*, LONG *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: return warning if failed

Definition at line 110 of file memory.c.

9.56.2.6 void fasp_mem_usage ()

Show total allocated memory currently.

Author

Chensong Zhang

Date

2010/08/12

Definition at line 169 of file memory.c.

9.56.3 Variable Documentation**9.56.3.1 unsigned INT total_alloc_count = 0**

total allocation times

Definition at line 33 of file memory.c.

9.56.3.2 unsigned INT total_alloc_mem = 0

total allocated memory

Definition at line 32 of file memory.c.

9.57 message.c File Reference

Output some useful messages.

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

Functions

- void [print_itinfo](#) (const [INT](#) ptrlvl, const [INT](#) stop_type, const [INT](#) iter, const [REAL](#) relres, const [REAL](#) absres, const [REAL](#) factor)
Print out iteration information for iterative solvers.
- void [print_amgcomplexity](#) ([AMG_data](#) *mgl, const [SHORT](#) 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.

9.57.1 Detailed Description

Output some useful messages.

Note

These routines are meant for internal use only.

Definition in file [message.c](#).

9.57.2 Function Documentation

9.57.2.1 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 195 of file message.c.

9.57.2.2 void void print_amgcomplexity (AMG_data * mgl, const SHORT prtlvl)

Print complexities of AMG method.

Parameters

<i>mgl</i>	Multilevel hierachy 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.

9.57.2.3 void void print_amgcomplexity_bsr (AMG_data_bsr * mgl, const SHORT prtlvl)

Print complexities of AMG method for BSR matrices.

Parameters

<i>mgl</i>	Multilevel hierachy for AMG
<i>prtlvl</i>	How much information to print

Author

Chensong Zhang

Date

05/10/2013

Definition at line 120 of file message.c.

9.57.2.4 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 161 of file message.c.

9.57.2.5 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.

9.57.2.6 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 178 of file message.c.

9.58 messages.h File Reference

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

Macros

- #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 `ERROR_OPEN_FILE` -10
Definition of return status and error messages.
- #define `ERROR_WRONG_FILE` -11
- #define `ERROR_INPUT_PAR` -13
- #define `ERROR_REGRESS` -14
- #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 `RUN_FAIL` -99
- #define `SUCCESS` 0
- #define `SOLVER_CG` 1

Definition of solver types for iterative methods.

- #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_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

Definition of solver types for direct methods (requires external libs)

- #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 [V_CYCLE](#) 1

Definition of cycle types.

- #define [W_CYCLE](#) 2
- #define [AML_I_CYCLE](#) 3
- #define [NL_AML_I_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_BLKOL](#) 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 [INTERP_DIR](#) 1

Definition of interpolation types.

- #define [INTERP_STD](#) 2
- #define [INTERP_ENG](#) 3
- #define [UNPT](#) -1

Type of vertices (dofs) for C/F splitting.

- #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

9.58.1 Detailed Description

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

Note

This is internal use only.

Created by Chensong Zhang on 03/20/2010. Modified by Chensong Zhang on 12/06/2011. Modified by Chensong Zhang on 12/25/2011. Modified by Chensong Zhang on 04/22/2012. Modified by Ludmil Zikatanov on 02/15/2013: CG -> SMOOTHER_CG. Modified by Chensong Zhang on 02/16/2013: GS -> SMOOTHER_GS, etc. Modified by Chensong Zhang on 04/09/2013: Add safe krylov methods.

Modified by Chensong Zhang on 09/22/2013: Clean up Doxygen.

Definition in file [messages.h](#).

9.58.2 Macro Definition Documentation

9.58.2.1 #define AMLI_CYCLE 3

AMLI-cycle

Definition at line 160 of file messages.h.

9.58.2.2 #define ASCEND 12

Asscending order

Definition at line 218 of file messages.h.

9.58.2.3 #define CF_ORDER 1

C/F order smoothing

Definition at line 210 of file messages.h.

9.58.2.4 #define CGPT 1

coarse grid points

Definition at line 203 of file messages.h.

9.58.2.5 #define CLASSIC_AMG 1

Definition of AMG types.

classic AMG

Definition at line 151 of file messages.h.

9.58.2.6 #define COARSE_AC 4

Aggressive coarsening

Definition at line 189 of file messages.h.

9.58.2.7 #define COARSE_CR 3

Compatible relaxation

Definition at line 188 of file messages.h.

9.58.2.8 #define COARSE_RS 1

Definition of coarsening types.

Classical coarsening

Definition at line 187 of file messages.h.

9.58.2.9 #define CPFIRST 1

C-points first order

Definition at line 216 of file messages.h.

9.58.2.10 #define DESCEND 21

Dsscending order

Definition at line 219 of file messages.h.

9.58.2.11 #define ERROR_ALLOC_MEM -20

fail to allocate memory

Definition at line 67 of file messages.h.

9.58.2.12 #define ERROR_AMG_COARSE_TYPE -32

unknown coarsening type

Definition at line 74 of file messages.h.

9.58.2.13 #define ERROR_AMG_COARSEING -33

coarsening step failed to complete

Definition at line 75 of file messages.h.

9.58.2.14 #define ERROR_AMG_INTERP_TYPE -30

unknown interpolation type

Definition at line 72 of file messages.h.

9.58.2.15 #define ERROR_AMG_SMOOTH_TYPE -31

unknown smoother type

Definition at line 73 of file messages.h.

9.58.2.16 #define ERROR_DATA_STRUCTURE -21

matrix or vector structures

Definition at line 68 of file messages.h.

9.58.2.17 #define ERROR_DATA_ZERODIAG -22

matrix has zero diagonal entries

Definition at line 69 of file messages.h.

9.58.2.18 #define ERROR_DUMMY_VAR -23

unexpected input data

Definition at line 70 of file messages.h.

9.58.2.19 #define ERROR_INPUT_PAR -13

wrong input argument

Definition at line 62 of file messages.h.

9.58.2.20 #define ERROR_LIC_TYPE -80

wrong license type

Definition at line 90 of file messages.h.

9.58.2.21 #define ERROR_MISC -19

other error

Definition at line 65 of file messages.h.

9.58.2.22 #define ERROR_NUM_BLOCKS -18

wrong number of blocks

Definition at line 64 of file messages.h.

9.58.2.23 #define ERROR_OPEN_FILE -10

Definition of return status and error messages.

fail to open a file

Definition at line 60 of file messages.h.

9.58.2.24 `#define ERROR_QUAD_DIM -61`

unsupported quadrature dim

Definition at line 88 of file messages.h.

9.58.2.25 `#define ERROR_QUAD_TYPE -60`

unknown quadrature type

Definition at line 87 of file messages.h.

9.58.2.26 `#define ERROR_REGRESS -14`

regression test fail

Definition at line 63 of file messages.h.

9.58.2.27 `#define ERROR_SOLVER_EXIT -49`

solver does not quit successfully

Definition at line 85 of file messages.h.

9.58.2.28 `#define ERROR_SOLVER_ILUSETUP -45`

ILU setup error

Definition at line 82 of file messages.h.

9.58.2.29 `#define ERROR_SOLVER_MAXIT -48`

maximal iteration number exceeded

Definition at line 84 of file messages.h.

9.58.2.30 `#define ERROR_SOLVER_MISC -46`

misc solver error during run time

Definition at line 83 of file messages.h.

9.58.2.31 `#define ERROR_SOLVER_PRECTYPE -41`

unknow precond type

Definition at line 78 of file messages.h.

9.58.2.32 #define ERROR_SOLVER_SOLSTAG -43

solver's solution is too small

Definition at line 80 of file messages.h.

9.58.2.33 #define ERROR_SOLVER_STAG -42

solver stagnates

Definition at line 79 of file messages.h.

9.58.2.34 #define ERROR_SOLVER_TOLSMALL -44

solver's tolerance is too small

Definition at line 81 of file messages.h.

9.58.2.35 #define ERROR_SOLVER_TYPE -40

unknown solver type

Definition at line 77 of file messages.h.

9.58.2.36 #define ERROR_WRONG_FILE -11

input contains wrong format

Definition at line 61 of file messages.h.

9.58.2.37 #define FALSE 0

logic FALSE

Definition at line 27 of file messages.h.

9.58.2.38 #define FGPT 0

fine grid points

Definition at line 202 of file messages.h.

9.58.2.39 #define FPFIRST -1

F-points first order

Definition at line 217 of file messages.h.

9.58.2.40 #define ILUk 1

Type of ILU methods.

ILUk

Definition at line 144 of file messages.h.

9.58.2.41 `#define ILUt 2`

ILUt

Definition at line 145 of file messages.h.

9.58.2.42 `#define ILUtp 3`

ILUtp

Definition at line 146 of file messages.h.

9.58.2.43 `#define INTERP_DIR 1`

Definition of interpolation types.

Direct interpolation

Definition at line 194 of file messages.h.

9.58.2.44 `#define INTERP_ENG 3`

energy minimization interp in C

Definition at line 196 of file messages.h.

9.58.2.45 `#define INTERP_STD 2`

Standard interpolation

Definition at line 195 of file messages.h.

9.58.2.46 `#define ISPT 2`

isolated points

Definition at line 204 of file messages.h.

9.58.2.47 `#define MAT_bBSR 5`

block matrix of BSR for bordered systems

Definition at line 53 of file messages.h.

9.58.2.48 `#define MAT_bCSR 4`

block matrix of CSR

Definition at line 52 of file messages.h.

9.58.2.49 #define MAT_BSR 2

blockwise compressed sparse row

Definition at line 50 of file messages.h.

9.58.2.50 #define MAT_CSR 1

compressed sparse row

Definition at line 49 of file messages.h.

9.58.2.51 #define MAT_CSRL 6

modified CSR to reduce cache missing

Definition at line 54 of file messages.h.

9.58.2.52 #define MAT_FREE 0

Definition of matrix format.

matrix-free format: only mxv action

Definition at line 48 of file messages.h.

9.58.2.53 #define MAT_STR 3

structured sparse matrix

Definition at line 51 of file messages.h.

9.58.2.54 #define MAT_SymCSR 7

symmetric CSR format

Definition at line 55 of file messages.h.

9.58.2.55 #define NL_AMLI_CYCLE 4

Nonlinear AMLI-cycle

Definition at line 161 of file messages.h.

9.58.2.56 #define NO_ORDER 0

Definition of smoothing order.

Natural order smoothing

Definition at line 209 of file messages.h.

9.58.2.57 #define OFF 0

turn off certain parameter

Definition at line 33 of file messages.h.

9.58.2.58 #define ON 1

Definition of switch.

turn on certain parameter

Definition at line 32 of file messages.h.

9.58.2.59 #define PREC_AMG 2

with AMG precondition

Definition at line 136 of file messages.h.

9.58.2.60 #define PREC_DIAG 1

with diagonal precondition

Definition at line 135 of file messages.h.

9.58.2.61 #define PREC_FMG 3

with full AMG precondition

Definition at line 137 of file messages.h.

9.58.2.62 #define PREC_ILU 4

with ILU precondition

Definition at line 138 of file messages.h.

9.58.2.63 #define PREC_NULL 0

Definition of preconditioner type for iterative methods.

with no precondition

Definition at line 134 of file messages.h.

9.58.2.64 #define PREC_SCHWARZ 5

with Schwarz preconditioner

Definition at line 139 of file messages.h.

9.58.2.65 #define PRINT_ALL 10

everything: all printouts allowed

Definition at line 43 of file messages.h.

9.58.2.66 #define PRINT_MIN 1

quiet: minimal print, like convergence

Definition at line 39 of file messages.h.

9.58.2.67 #define PRINT_MORE 4

more: print more useful information

Definition at line 41 of file messages.h.

9.58.2.68 #define PRINT_MOST 8

most: maximal printouts, no disk files

Definition at line 42 of file messages.h.

9.58.2.69 #define PRINT_NONE 0

Print level for all subroutines – not including DEBUG output.

silent: no printout at all

Definition at line 38 of file messages.h.

9.58.2.70 #define PRINT_SOME 2

some: print cpu time, iteration number

Definition at line 40 of file messages.h.

9.58.2.71 #define RUN_FAIL -99

general failure

Definition at line 92 of file messages.h.

9.58.2.72 #define SA_AMG 2

smoothed aggregation AMG

Definition at line 152 of file messages.h.

9.58.2.73 #define SMOOTHER_BLKOIL 11

Definition of specialized smoother types.
Used in monolithic AMG for black-oil
Definition at line 180 of file messages.h.

9.58.2.74 #define SMOOTHER_CG 4

CG as a smoother
Definition at line 169 of file messages.h.

9.58.2.75 #define SMOOTHER_GS 2

Gauss-Seidel smoother
Definition at line 167 of file messages.h.

9.58.2.76 #define SMOOTHER_GSOR 7

GS + SOR smoother
Definition at line 172 of file messages.h.

9.58.2.77 #define SMOOTHER_JACOBI 1

Definition of standard smoother types.
Jacobi smoother
Definition at line 166 of file messages.h.

9.58.2.78 #define SMOOTHER_L1DIAG 10

L1 norm diagonal scaling smoother
Definition at line 175 of file messages.h.

9.58.2.79 #define SMOOTHER_POLY 9

Polynomial smoother
Definition at line 174 of file messages.h.

9.58.2.80 #define SMOOTHER_SGS 3

symm Gauss-Seidel smoother
Definition at line 168 of file messages.h.

9.58.2.81 #define SMOOTHER_SGSOR 8

SGS + SSOR smoother

Definition at line 173 of file messages.h.

9.58.2.82 #define SMOOTHER_SOR 5

SOR smoother

Definition at line 170 of file messages.h.

9.58.2.83 #define SMOOTHER_SPETEN 19

Used in monolithic AMG for black-oil

Definition at line 181 of file messages.h.

9.58.2.84 #define SMOOTHER_SSOR 6

SSOR smoother

Definition at line 171 of file messages.h.

9.58.2.85 #define SOLVER_AMG 21

AMG as an iterative solver

Definition at line 114 of file messages.h.

9.58.2.86 #define SOLVER_BiCGstab 2

Biconjugate Gradient Stabilized

Definition at line 99 of file messages.h.

9.58.2.87 #define SOLVER_CG 1

Definition of solver types for iterative methods.

Conjugate Gradient

Definition at line 98 of file messages.h.

9.58.2.88 #define SOLVER_FMG 22

Full AMG as an solver

Definition at line 115 of file messages.h.

9.58.2.89 #define SOLVER_GCG 7

Generalized Conjugate Gradient

Definition at line 104 of file messages.h.

9.58.2.90 #define SOLVER_GMRES 4

Generalized Minimal Residual

Definition at line 101 of file messages.h.

9.58.2.91 #define SOLVER_MinRes 3

Minimal Residual

Definition at line 100 of file messages.h.

9.58.2.92 #define SOLVER_MUMPS 33

MUMPS Direct Solver

Definition at line 122 of file messages.h.

9.58.2.93 #define SOLVER_SBiCGstab 12

BiCGstab with safe net

Definition at line 107 of file messages.h.

9.58.2.94 #define SOLVER_SCG 11

Conjugate Gradient with safe net

Definition at line 106 of file messages.h.

9.58.2.95 #define SOLVER_SGCG 17

GCG with safe net

Definition at line 112 of file messages.h.

9.58.2.96 #define SOLVER_SGMRES 14

GMRes with safe net

Definition at line 109 of file messages.h.

9.58.2.97 #define SOLVER_SMinRes 13

MinRes with safe net

Definition at line 108 of file messages.h.

9.58.2.98 #define SOLVER_SUPERLU 31

Definition of solver types for direct methods (requires external libs)

SuperLU Direct Solver

Definition at line 120 of file messages.h.

9.58.2.99 #define SOLVER_SVFGMRES 16

Variable-restart FGMRES with safe net

Definition at line 111 of file messages.h.

9.58.2.100 #define SOLVER_SVGMRES 15

Variable-restart GMRES with safe net

Definition at line 110 of file messages.h.

9.58.2.101 #define SOLVER_UMFPACK 32

UMFPack Direct Solver

Definition at line 121 of file messages.h.

9.58.2.102 #define SOLVER_VFGMRES 6

Variable Restarting Flexible GMRES

Definition at line 103 of file messages.h.

9.58.2.103 #define SOLVER_VGMRES 5

Variable Restarting GMRES

Definition at line 102 of file messages.h.

9.58.2.104 #define STOP_MOD_REL_RES 3

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

Definition at line 129 of file messages.h.

9.58.2.105 #define STOP_REL_PRECRES 2

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

Definition at line 128 of file messages.h.

9.58.2.106 #define STOP_REL_RES 1

Definition of iterative solver stopping criteria types.

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

Definition at line 127 of file messages.h.

9.58.2.107 #define SUCCESS 0

return from function successfully

Definition at line 93 of file messages.h.

9.58.2.108 #define TRUE 1

Definition of logic type.

logic TRUE

Definition at line 26 of file messages.h.

9.58.2.109 #define UA_AMG 3

unsmoothed aggregation AMG

Definition at line 153 of file messages.h.

9.58.2.110 #define UNPT -1

Type of vertices (dofs) for C/F splitting.

undetermined points

Definition at line 201 of file messages.h.

9.58.2.111 #define USERDEFINED 0

Type of ordering for smoothers.

USERDEFINED order

Definition at line 215 of file messages.h.

9.58.2.112 #define V_CYCLE 1

Definition of cycle types.

V-cycle

Definition at line 158 of file messages.h.

9.58.2.113 #define W_CYCLE 2

W-cycle

Definition at line 159 of file messages.h.

9.59 mg_util.inl File Reference

Routines for algebraic multigrid cycles.

9.59.1 Detailed Description

Routines for algebraic multigrid cycles.

Definition in file [mg_util.inl](#).

9.60 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"
```

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.

9.60.1 Detailed Description

Abstract non-recursive multigrid cycle.

Definition in file [mgcycle.c](#).

9.60.2 Function Documentation

9.60.2.1 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.

Definition at line 41 of file mgcycle.c.

9.60.2.2 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 294 of file mgcycle.c.

9.61 mgrecur.c File Reference

Abstract multigrid cycle – recursive version.

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

Functions

- void [fasp_solver_mgrecur](#) ([AMG_data](#) **mgl*, [AMG_param](#) **param*, [INT](#) level)

Solve $Ax=b$ with recursive multigrid K -cycle.

9.61.1 Detailed Description

Abstract multigrid cycle – recursive version.

Definition in file [mgrecur.c](#).

9.61.2 Function Documentation

9.61.2.1 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 31 of file mgrecur.c.

9.62 ordering.c File Reference

A collection of ordering, merging, removing duplicated integers functions.

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

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_msort](#) (INT numbers[], INT work[], INT left, INT right)
Sort the INT array ascendingly with the merge sort algorithm.
- void [fasp_aux_iQuickSort](#) (INT *a, INT left, INT right)
Sort the array (INT type) ascendingly with the quick sorting algorithm.
- void [fasp_aux_dQuickSort](#) (REAL *a, INT left, INT right)
Sort the array (REAL type) ascendingly 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.

9.62.1 Detailed Description

A collection of ordering, merging, removing duplicated integers functions.

Definition in file [ordering.c](#).

9.62.2 Function Documentation

9.62.2.1 void `fasp_aux_dQuickSort` (`REAL * a`, `INT left`, `INT right`)

Sort the array (REAL type) ascendingly 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 238 of file `ordering.c`.

9.62.2.2 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 319 of file ordering.c.

9.62.2.3 void fasp_aux_iQuickSort (INT * a, INT left, INT right)

Sort the array (INT type) ascendingly 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 200 of file ordering.c.

9.62.2.4 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 278 of file ordering.c.

9.62.2.5 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 107 of file ordering.c.

9.62.2.6 void fasp_aux_msort (INT *numbers*[], INT *work*[], INT *left*, INT *right*)

Sort the INT array ascendingly 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 169 of file ordering.c.

9.62.2.7 `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 74 of file ordering.c.

9.62.2.8 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 29 of file ordering.c.

9.63 parameters.c File Reference

Initialize, set, or print input data and parameters.

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

Functions

- void [fasp_param_set](#) (int argc, const char *argv[], [input_param](#) *iniparam)
Read input from command-line arguments.
- void [fasp_param_init](#) ([input_param](#) *iniparam, [itsolver_param](#) *itsparam, [AMG_param](#) *amgparam, [ILU_param](#) *iluparam, [Schwarz_param](#) *schparam)
Initialize parameters, global variables, etc.
- void [fasp_param_input_init](#) ([input_param](#) *iniparam)
Initialize input parameters.
- void [fasp_param_amg_init](#) ([AMG_param](#) *amgparam)
Initialize AMG parameters.
- void [fasp_param_solver_init](#) ([itsolver_param](#) *itsparam)
Initialize [itsolver_param](#).
- void [fasp_param_ilu_init](#) ([ILU_param](#) *iluparam)
Initialize ILU parameters.
- void [fasp_param_schwarz_init](#) ([Schwarz_param](#) *schparam)
Initialize Schwarz parameters.
- void [fasp_param_amg_set](#) ([AMG_param](#) *param, [input_param](#) *iniparam)
Set [AMG_param](#) from INPUT.
- void [fasp_param_ilu_set](#) ([ILU_param](#) *iluparam, [input_param](#) *iniparam)
Set [ILU_param](#) with INPUT.
- void [fasp_param_schwarz_set](#) ([Schwarz_param](#) *schparam, [input_param](#) *iniparam)
Set [Schwarz_param](#) with INPUT.
- void [fasp_param_solver_set](#) ([itsolver_param](#) *itsparam, [input_param](#) *iniparam)
Set [itsolver_param](#) with INPUT.
- void [fasp_param_amg_to_prec](#) ([precond_data](#) *pcdata, [AMG_param](#) *amgparam)
Set [precond_data](#) with [AMG_param](#).
- void [fasp_param_prec_to_amg](#) ([AMG_param](#) *amgparam, [precond_data](#) *pcdata)
Set [AMG_param](#) with [precond_data](#).
- void [fasp_param_amg_to_prec_bsr](#) ([precond_data_bsr](#) *pcdata, [AMG_param](#) *amgparam)
Set [precond_data_bsr](#) with [AMG_param](#).
- void [fasp_param_prec_to_amg_bsr](#) ([AMG_param](#) *amgparam, [precond_data_bsr](#) *pcdata)
Set [AMG_param](#) with [precond_data](#).
- void [fasp_param_amg_print](#) ([AMG_param](#) *param)
Print out AMG parameters.
- void [fasp_param_ilu_print](#) ([ILU_param](#) *param)
Print out ILU parameters.
- void [fasp_param_schwarz_print](#) ([Schwarz_param](#) *param)
Print out Schwarz parameters.
- void [fasp_param_solver_print](#) ([itsolver_param](#) *param)
Print out itsolver parameters.

9.63.1 Detailed Description

Initialize, set, or print input data and parameters.

Definition in file [parameters.c](#).

9.63.2 Function Documentation

9.63.2.1 void fasp_param_amg_init (AMG_param * *amgparam*)

Initialize AMG parameters.

Parameters

<i>amgparam</i>	Parameters for AMG
-----------------	--------------------

Author

Chensong Zhang

Date

2010/04/03

Definition at line 385 of file parameters.c.

9.63.2.2 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 776 of file parameters.c.

9.63.2.3 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 507 of file parameters.c.

9.63.2.4 void fasp_param_amg_to_prec (precondition_data * pcddata, 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 649 of file parameters.c.

9.63.2.5 void fasp_param_amg_to_prec_bsr (precondition_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 714 of file parameters.c.

9.63.2.6 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 468 of file parameters.c.

9.63.2.7 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 869 of file parameters.c.

9.63.2.8 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 577 of file parameters.c.

9.63.2.9 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.

9.63.2.10 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.

9.63.2.11 void fasp_param_prec_to_amg (AMG_param * *amgparam*, precondition_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 683 of file parameters.c.

9.63.2.12 void fasp_param_prec_to_amg_bsr (AMG_param * *amgparam*, precondition_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 747 of file parameters.c.

9.63.2.13 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

Definition at line 488 of file parameters.c.

9.63.2.14 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 899 of file parameters.c.

9.63.2.15 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 599 of file parameters.c.

9.63.2.16 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.

9.63.2.17 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 447 of file parameters.c.

9.63.2.18 void fasp_param_solver_print (itsolver_param * param)

Print out itsolver parameters.

Parameters

<i>param</i>	Parameters for iterative solvers
--------------	----------------------------------

Author

Chensong Zhang

Date

2011/12/20

Definition at line 928 of file parameters.c.

9.63.2.19 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 619 of file parameters.c.

9.64 pbcgs.c File Reference

Krylov subspace methods – Preconditioned BiCGstab.

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

Functions

- `INT fasp_solver_dcsr_pbcgs (dCSRmat *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 $Au=b$.
- `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 $Au=b$.
- `INT fasp_solver_bdcsr_pbcgs (block_dCSRmat *A, dvector *b, dvector *u, precondition *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, precondition *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT print_level)`
Preconditioned BiCGstab method for solving $Au=b$.

9.64.1 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: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

Definition in file [pbcgs.c](#).

9.64.2 Function Documentation

9.64.2.1 **INT fasp_solver_bdcsr_pbcgs (block_dCSRmat * A, dvector * b, dvector * u, precondition * 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 precondition: 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 781 of file pbcgs.c.

9.64.2.2 `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 precondition: 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 435 of file pbcgs.c.

9.64.2.3 `INT fasp_solver_dcsr_pbcgs (dCSRmat * 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 precondition: 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 89 of file pbcgs.c.

9.64.2.4 `INT fasp_solver_dstr_pbcgs (dSTRmat * 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 precondition: 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 1127 of file pbcgs.c.

9.65 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"
```

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 $Au=b$.

9.65.1 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 $k = 0:MaxIt$

- 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

Definition in file [pbcgs_mf.c](#).

9.65.2 Function Documentation

9.65.2.1 **INT fasp_solver_pbcgs (mxv_matfree * mf, 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>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 precondition: 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.

9.66 pcg.c File Reference

Krylov subspace methods – Preconditioned conjugate gradient.

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

Functions

- [INT fasp_solver_dcsr_pcg](#) ([dCSRmat](#) *A, [dvector](#) *b, [dvector](#) *u, [precond](#) *pc, const [REAL](#) tol, const [INT](#) MaxIt, const [SHORT](#) stop_type, const [SHORT](#) 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 $Au=b$.

9.66.1 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

Definition in file [pcg.c](#).

9.66.2 Function Documentation

9.66.2.1 `INT fasp_solver_bdcsr_pcg (block_dCSRmat * 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 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 precondition: 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 367 of file pcg.c.

9.66.2.2 `INT fasp_solver_dcsr_pcg (dCSRmat * 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 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 precondition: 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.

9.66.2.3 **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 precondition: 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 648 of file pcg.c.

9.67 pcg_mf.c File Reference

Krylov subspace methods – Preconditioned conjugate gradient (matrix free)

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

Functions

- **INT** fasp_solver_pcg (**mxv_matfree** **mf*, **dvector** **b*, **dvector** **u*, **precond** **pc*, **const REAL** *tol*, **const INT** *MaxIt*, **const SHORT** *stop_type*, **const SHORT** *print_level*)
Preconditioned conjugate gradient (CG) method for solving $Au=b$.

9.67.1 Detailed Description

Krylov subspace methods – Preconditioned conjugate gradient (matrix free) Abstract algorithm

PCG method to solve $Ax=b$ is to generate $\{x_k\}$ to approximate x

Step 0. Given A, b, x_0, M

Step 1. Compute residual $r_0 = b - Ax_0$ 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: $r_{k+1} = r_k - \alpha(Ap_k)$;
- 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 $r = b - Ax_{k+1}$;
 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 $r = b - Ax_{k+1}$;
 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

Definition in file [pcg_mf.c](#).

9.67.2 Function Documentation

9.67.2.1 **INT** fasp_solver_pcg (mxv_matfree * mf, dvector * b, dvector * u, precondition * 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 precondition: 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`.

9.68 pgcg.c File Reference

Krylov subspace methods – Preconditioned Generalized CG.

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

Functions

- [INT fasp_solver_dcsr_pgcg](#) ([dCSRmat](#) *A, [dvector](#) *b, [dvector](#) *u, [precond](#) *pc, const [REAL](#) tol, const [INT](#) MaxIt, const [SHORT](#) stop_type, const [SHORT](#) print_level)

Preconditioned generalized conjugate gradient (GCG) method for solving $Au=b$.

9.68.1 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

Definition in file [pgcg.c](#).

9.68.2 Function Documentation

9.68.2.1 **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 precondition: 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.

9.69 pgcg_mf.c File Reference

Krylov subspace methods – Preconditioned Generalized CG (matrix free)

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

Functions

- **INT** fasp_solver_pgcg (**mxv_matfree** **mf*, **dvector** **b*, **dvector** **u*, **precond** **pc*, const **REAL** *tol*, const **INT** *MaxIt*, const **SHORT** *stop_type*, const **SHORT** *print_level*)

Preconditioned generalized conjugate gradient (GCG) method for solving $Au=b$.

9.69.1 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

Definition in file [pgcg_mf.c](#).

9.69.2 Function Documentation

9.69.2.1 `INT fasp_solver_pgcg (mxv_matfree * mf, dvector * b, dvector * u, precondition * 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 precondition: 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](#).

9.70 pgmres.c File Reference

Krylov subspace methods – Preconditioned GMRes.

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

Functions

- [INT fasp_solver_dcsr_pgmres](#) ([dCSRmat](#) *A, [dvector](#) *b, [dvector](#) *x, [precond](#) *pc, const [REAL](#) tol, const [INT](#) MaxIt, [SHORT](#) restart, const [SHORT](#) stop_type, const [SHORT](#) print_level)
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, [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, [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, [SHORT](#) restart, const [SHORT](#) stop_type, const [SHORT](#) print_level)
Preconditioned GMRES method for solving $Au=b$.

9.70.1 Detailed Description

Krylov subspace methods – Preconditioned GMRes.

Note

Refer to Y. Saad 2003 Iterative methods for sparse linear systems (2nd Edition), SIAM
See also [pvgmres.c](#) for a variable restarting version.
See [spgmres.c](#) for a safer version

Definition in file [pgmres.c](#).

9.70.2 Function Documentation

9.70.2.1 [INT fasp_solver_bdcsr_pgmres](#) ([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

A	Pointer to block_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 precondition: 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 345 of file pgmres.c.

9.70.2.2 `INT fasp_solver_dbsr_pgmres (dBSRmat * A, dvector * b, dvector * x, precondition * 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 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 precondition: 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 641 of file pgmres.c.

9.70.2.3 `INT fasp_solver_dcsr_pgmres (dCSRmat * A, dvector * b, dvector * x, precondition * 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 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 precondition: 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

Definition at line 49 of file pgmres.c.

9.70.2.4 `INT fasp_solver_dstr_pgmres (dSTRmat * A, dvector * b, dvector * x, precondition * 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 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 precondition: 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 937 of file pgmres.c.

9.71 pgmres_mf.c File Reference

Krylov subspace methods – Preconditioned GMRes (matrix free)

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

Functions

- [INT fasp_solver_pgmres](#) ([mxv_matfree](#) *mf, [dvector](#) *b, [dvector](#) *x, [precond](#) *pc, const [REAL](#) tol, const [INT](#) MaxIt, [SHORT](#) restart, const [SHORT](#) stop_type, const [SHORT](#) print_level)
Solve "Ax=b" using PGMRES (right preconditioned) iterative method.

9.71.1 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 GMRES(m) Journal of Computational and Applied Mathematics, 230 (2009) pp. 751-761. UCRL-JRNL-235266.

Definition in file [pgmres_mf.c](#).

9.71.2 Function Documentation

9.71.2.1 **INT** fasp_solver_pgmres (**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.

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 precondition: 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.

9.72 pminres.c File Reference

Krylov subspace methods – Preconditioned minimal residual.

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

Functions

- **INT** fasp_solver_dcsr_pminres (**dCSRmat** **A*, **dvector** **b*, **dvector** **u*, **precond** **pc*, const **REAL** *tol*, const **INT** *MaxIt*, const **SHORT** *stop_type*, const **SHORT** *print_level*)
A preconditioned minimal residual (Minres) method for solving $Au=b$.

- `INT fasp_solver_bdcscr_pminres (block_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 $Au=b$.

- `INT fasp_solver_dstr_pminres (dSTRmat *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 $Au=b$.

9.72.1 Detailed Description

Krylov subspace methods – Preconditioned minimal residual. Abstract algorithm of Krylov method

Krylov method to solve $Ax=b$ 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: $\text{norm}(r)/\text{norm}(b) < \text{tol}$

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

Definition in file [pminres.c](#).

9.72.2 Function Documentation

9.72.2.1 **INT fasp_solver_bdcsr_pminres (block_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 $Au=b$.

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 precondition: 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 499 of file [pminres.c](#).

9.72.2.2 **INT fasp_solver_dcsr_pminres (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 $Au=b$.

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 precondition: 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.

9.72.2.3 `INT fasp_solver_dstr_pminres (dSTRmat * 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 $Au=b$.

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 precondition: 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 901 of file pminres.c.

9.73 pminres_mf.c File Reference

Krylov subspace methods – Preconditioned minimal residual (matrix free)

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

Functions

- `INT fasp_solver_pminres (mxv_matfree *mf, dvector *b, dvector *u, precondition *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT print_level)`

A preconditioned minimal residual (Minres) method for solving $Au=b$.

9.73.1 Detailed Description

Krylov subspace methods – Preconditioned minimal residual (matrix free) Abstract algorithm of Krylov method

Krylov method to solve $A*x=b$ is to generate $\{x_k\}$ to approximate x , where x_k is the optimal solution in Krylov space

$V_k = \text{span}\{r_0, A*r_0, A^2*r_0, \dots, A^{k-1}*r_0\}$,

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

Definition in file [pminres_mf.c](#).

9.73.2 Function Documentation

9.73.2.1 `INT fasp_solver_pminres (mxv_matfree * mf, 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 $Au=b$.

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 precondition: 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.

9.74 precondition_bsr.c File Reference

Preconditioners for [dBSRmat](#) matrices.

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

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.

9.74.1 Detailed Description

Preconditioners for [dBSRmat](#) matrices.

Definition in file [precond_bsr.c](#).

9.74.2 Function Documentation

9.74.2.1 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 561 of file precondition_bsr.c.

9.74.2.2 void fasp_precond_dbsr_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

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 35 of file precondition_bsr.c.

9.74.2.3 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 109 of file precondition_bsr.c.

9.74.2.4 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 159 of file precondition_bsr.c.

9.74.2.5 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 209 of file `precond_bsr.c`.

9.74.2.6 `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 258 of file `precond_bsr.c`.

9.74.2.7 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

Date

11/09/2010

Note

Works for general nb (Xiaozhe)

Definition at line 304 of file `precond_bsr.c`.

9.74.2.8 `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 604 of file `precond_bsr.c`.

9.75 `precond_csr.c` File Reference

Preconditioners for [dCSRmat](#) matrices.

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

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 $z = \text{inv}(D) * r$.*
- `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_free` (`SHORT` `precond_type`, `precond` *`pc`)
free preconditioner

9.75.1 Detailed Description

Preconditioners for `dCSRmat` matrices.

Definition in file `precond_csr.c`.

9.75.2 Function Documentation

9.75.2.1 `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 417 of file `precond_csr.c`.

9.75.2.2 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 488 of file precondition_csr.c.

9.75.2.3 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 165 of file precondition_csr.c.

9.75.2.4 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 455 of file precondition_csr.c.

9.75.2.5 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 557 of file precondition_csr.c.

9.75.2.6 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 191 of file precondition_csr.c.

9.75.2.7 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 313 of file `precond_csr.c`.

9.75.2.8 `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, Shiquan Zhang

Date

04/06/2010

Definition at line 260 of file `precond_csr.c`.

9.75.2.9 `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 521 of file `precond_csr.c`.

9.75.2.10 void fasp_precond_schwarz(REAL * *r*, REAL * *z*, void * *data*)

get *z* from *r* by schwarz

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

03/22/2010

Definition at line 363 of file `precond_csr.c`.

9.75.2.11 `precond * fasp_precond_setup (SHORT precond_type, AMG_param * amgparam, ILU_param * iluparam, dCSRmat * A)`

Setup preconditioner interface for iterative methods.

Parameters

<i>precond_type</i>	Preconditioner type
<i>*amgparam</i>	AMG parameters
<i>*iluparam</i>	ILU parameters
<i>*A</i>	Pointer to coefficient matrix

Returns

Pointer to preconditioner

Author

Feiteng Huang

Date

05/18/2009

Definition at line 30 of file `precond_csr.c`.

9.76 `precond_str.c` File Reference

Preconditioners for `dSTRmat` matrices.

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

Functions

- void `fasp_precond_dstr_diag` (`REAL *r`, `REAL *z`, void *data)
*Diagonal preconditioner $z = \text{inv}(D) * r$.*
- 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)

9.76.1 Detailed Description

Preconditioners for `dSTRmat` matrices.

Definition in file `precond_str.c`.

9.76.2 Function Documentation

9.76.2.1 void fasp_precond_dstr_blockgs (REAL * r, REAL * z, void * data)

CPR-type preconditioner (STR format)

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

10/17/2010

Definition at line 1707 of file `precond_str.c`.

9.76.2.2 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 precondition_str.c.

9.76.2.3 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.

9.76.2.4 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
<i>data</i>	Pointer to precondition data

Author

Shiquan Zhang

Date

06/07/2010

Definition at line 979 of file precondition_str.c.

9.76.2.5 void fasp_precond_dstr_ilu0_forward (REAL * *r*, REAL * *z*, void * *data*)

Preconditioning using STR_ILU(0) 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

06/07/2010

Definition at line 816 of file precondition_str.c.

9.76.2.6 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 precondition_str.c.

9.76.2.7 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.

9.76.2.8 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`.

9.77 pvfgmres.c File Reference

Krylov subspace methods – Preconditioned variable-restarting flexible GMRes.

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

Functions

- [INT fasp_solver_dcsr_pvfgmres](#) ([dCSRmat](#) *A, [dvector](#) *b, [dvector](#) *x, [precond](#) *pc, const [REAL](#) tol, const [INT](#) MaxIt, [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, [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.

9.77.1 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 GMRES(m) Journal of Computational and Applied Mathematics, 230 (2009) pp. 751-761. UCRL-JRNL-235266.
This file is modified from [pvgmres.c](#)

Definition in file [pvfgmres.c](#).

9.77.2 Function Documentation

9.77.2.1 **INT** fasp_solver_dbsr_pvfgmres (**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 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 precondition: 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 373 of file pvfgmres.c.

9.77.2.2 `INT fasp_solver_dcsr_pvfgmres (dCSRmat * A, dvector * b, dvector * x, precondition * pc, const REAL tol, const INT MaxIt, 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 precondition: 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.

9.78 pvfgmres_mf.c File Reference

Krylov subspace methods – Preconditioned variable-restarting flexible GMRes (matrix free)

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

Functions

- [INT fasp_solver_pvfgmres](#) ([mxv_matfree](#) *mf, [dvector](#) *b, [dvector](#) *x, [precond](#) *pc, const [REAL](#) tol, const [INT](#) MaxIt, [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.

9.78.1 Detailed Description

Krylov subspace methods – Preconditioned variable-restarting flexible GMRes (matrix free)

Note

Refer to Y. Saad 2003 Iterative methods for sparse linear systems (2nd Edition), SIAM

Refer to A.H. Baker, E.R. Jessup, and Tz.V. Kolev A Simple Strategy for Varying the Restart Parameter in GMRES(m) Journal of Computational and Applied Mathematics, 230 (2009) pp. 751-761. UCRL-JRNL-235266.

This file is modified from [pvgmres.c](#)

Definition in file [pvfgmres_mf.c](#).

9.78.2 Function Documentation

9.78.2.1 [INT fasp_solver_pvfgmres](#) ([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 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 precondition: 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.

9.79 pvgmres.c File Reference

Krylov subspace methods – Preconditioned variable-restart GMRes.

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

Functions

- [INT fasp_solver_dcsr_pvgmres](#) ([dCSRmat](#) *A, [dvector](#) *b, [dvector](#) *x, [precond](#) *pc, const [REAL](#) tol, const [INT](#) MaxIt, [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_pvgmres](#) ([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)
Solve "Ax=b" using PGMRES(right preconditioned) iterative 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, [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_pvgmres (dSTRmat *A, dvector *b, dvector *x, precondition *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.

9.79.1 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

Definition in file [pvgmres.c](#).

9.79.2 Function Documentation

- 9.79.2.1 `INT fasp_solver_bdcsr_pvgmres (block_dCSRmat * A, dvector * b, dvector * x, precondition * 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 precondition: 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 386 of file [pvgmres.c](#).

9.79.2.2 **INT** fasp_solver_dbsr_pvgmres (**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 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 precondition: 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 723 of file pvgmres.c.

9.79.2.3 `INT fasp_solver_dcsr_pvgmres (dCSRmat * A, dvector * b, dvector * x, precondition * 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 precondition: 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.

9.79.2.4 `INT fasp_solver_dstr_pvgmres (dSTRmat * A, dvector * b, dvector * x, precondition * 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 precondition: 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 1060 of file pvgmres.c.

9.80 pvgmres_mf.c File Reference

Krylov subspace methods – Preconditioned variable-restarting GMRes (matrix free)

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


Functions

- **INT fasp_solver_pvgmres** (**mxv_matfree** *mf, **dvector** *b, **dvector** *x, **precond** *pc, const **REAL** tol, const **INT** MaxIt, **SHORT** restart, const **SHORT** stop_type, const **SHORT** print_level)
Solve "Ax=b" using PGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during the iteration.

9.80.1 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 GMRES(m) Journal of Computational and Applied Mathematics, 230 (2009) pp. 751-761. UCRL-JRNL-235266.

Definition in file [pvgmres_mf.c](#).

9.80.2 Function Documentation

9.80.2.1 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 precondition: 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.

9.81 quadrature.c File Reference

Quadrature rules.

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

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.

9.81.1 Detailed Description

Quadrature rules.

Definition in file [quadrature.c](#).

9.81.2 Function Documentation

9.81.2.1 void [fasp_gauss2d](#) (INT num_qp, INT ncoor, REAL(*) gauss[3])

Initialize Gauss quadrature points and weights.

Parameters

<i>num_qp</i>	Number of quadrature points
<i>ncoor</i>	Dimension of space
<i>gauss</i>	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](#).

9.81.2.2 void [fasp_quad2d](#) (INT num_qp, INT ncoor, REAL(*) quad[3])

Initialize Lagrange quadrature points and weights.

Parameters

<i>num_qp</i>	Number of quadrature points
<i>ncoor</i>	Dimension of space
<i>quad</i>	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.

9.82 rap.c File Reference

R*A*P driver.

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

Functions

- [dCSRmat fasp_blas_dcsr_rap2](#) ([INT](#) *ir, [INT](#) *jr, [REAL](#) *r, [INT](#) *ia, [INT](#) *ja, [REAL](#) *a, [INT](#) *ipt, [INT](#) *jpt, [REAL](#) *pt, [INT](#) n, [INT](#) nc, [INT](#) *maxrpout, [INT](#) *ipin, [INT](#) *jpin)
Compute $R*A*P$.

9.82.1 Detailed Description

R*A*P driver.

C-version by Ludmil Zikatanov 2010-04-08

tested 2010-04-08

Definition in file [rap.c](#).

9.82.2 Function Documentation

9.82.2.1 [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](#) *maxrpout, [INT](#) *ipin, [INT](#) *jpin)

Compute $R*A*P$.

Author

Ludmil Zikatanov

Date

04/08/2010

Note

It uses [dCSRmat](#) only. The functions called from here are in [sparse_util.c](#)

Definition at line 33 of file rap.c.

9.83 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)

9.83.1 Detailed Description

Schwarz smoothers.

Author

Ludmil Zikatanov

Date

01/01/2007

Note

These routines are part of the matching MG method

Definition in file [schwarz.f](#).

9.84 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"
```

Functions

- [INT fasp_schwarz_setup](#) ([Schwarz_data](#) *schwarz, [INT](#) mmsize, [INT](#) maxlev, [INT](#) schwarz_type)
Setup phase for the Schwarz methods.

9.84.1 Detailed Description

Setup phase for the Schwarz methods.

Definition in file [schwarz_setup.c](#).

9.84.2 Function Documentation

9.84.2.1 [INT fasp_schwarz_setup](#) ([Schwarz_data](#) * *schwarz*, [INT](#) *mmsize*, [INT](#) *maxlev*, [INT](#) *schwarz_type*)

Setup phase for the Schwarz methods.

Parameters

<i>schwarz</i>	Pointer to the showarz data
<i>mmsize</i>	Max block size
<i>maxlev</i>	Max number of levels
<i>schwarz_type</i>	Type of the Schwarz method

Returns

SUCCESS if succeed

Author

Ludmil, Xiaozhe Hu

Date

03/22/2011

Modified by Chunsheng Feng, Zheng Li on 08/28/2012 find the blocks

LU decomposition of blocks

return

Definition at line 35 of file schwarz_setup.c.

9.85 smat.c File Reference

Simple operations for *small* full matrices in row-major format.

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

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.*

9.85.1 Detailed Description

Simple operations for *small* full matrices in row-major format.

Definition in file [smat.c](#).

9.85.2 Function Documentation

9.85.2.1 `INT fasp_blas_smat_inv (REAL * a, const INT n)`

Compute the inverse matrix of a small full matrix a.

Parameters

<i>a</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 392 of file smat.c.

9.85.2.2 `void fasp_blas_smat_inv_nc2 (REAL * a)`

Compute the inverse matrix of a 2*2 full matrix A (in place)

Parameters

<i>a</i>	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.

9.85.2.3 `void fasp_blas_smat_inv_nc3 (REAL * a)`

Compute the inverse matrix of a 3*3 full matrix A (in place)

Parameters

<i>a</i>	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.

9.85.2.4 void fasp_blas_smat_inv_nc4 (REAL * *a*)

Compute the inverse matrix of a 4*4 full matrix A (in place)

Parameters

<i>a</i>	Pointer to the REAL array which stands a 4*4 matrix
----------	---

Author

Xiaozhe Hu

Date

01/12/2013

Definition at line 102 of file smat.c.

9.85.2.5 void fasp_blas_smat_inv_nc5 (REAL * *a*)

Compute the inverse matrix of a 5*5 full matrix A (in place)

Parameters

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

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 160 of file smat.c.

9.85.2.6 void fasp_blas_smat_inv_nc7 (REAL * *a*)

Compute the inverse matrix of a 7*7 matrix a.

Parameters

<i>a</i>	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 376 of file smat.c.

9.85.2.7 void fasp_iden_free (idenmat * *A*)

Free idenmat sparse matrix data memeory space.

Parameters

<i>A</i>	Pointer to the idenmat matrix
----------	-------------------------------

Author

Chensong Zhang

Date

2010/04/03

Definition at line 481 of file smat.c.

9.85.2.8 void fasp_smat_identity (REAL * *a*, INT *n*, INT *n2*)

Set a $n \times n$ full matrix to be a identity.

Parameters

<i>a</i>	Pointer to the REAL vector which stands for a $n \times n$ full matrix
<i>n</i>	Size of full matrix
<i>n2</i>	Length of the REAL vector which stores the $n \times n$ full matrix

Author

Xiaozhe Hu

Date

2010/12/25

Definition at line 581 of file smat.c.

9.85.2.9 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 501 of file smat.c.

9.85.2.10 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 518 of file smat.c.

9.85.2.11 void fasp_smat_identity_nc5 (REAL * *a*)

Set a 5*5 full matrix to be a identity.

Parameters

<i>a</i>	Pointer to the REAL vector which stands for a 5*5 full matrix
----------	---

Author

Xiaozhe Hu

Date

2010/12/25

Definition at line 535 of file smat.c.

9.85.2.12 void fasp_smat_identity_nc7 (REAL * *a*)

Set a 7*7 full matrix to be a identity.

Parameters

<i>a</i>	Pointer to the REAL vector which stands for a 7*7 full matrix
----------	---

Author

Xiaozhe Hu

Date

2010/12/25

Definition at line 556 of file smat.c.

9.86 smoother_bsr.c File Reference

Smoothers for [dBSRmat](#) matrices.

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

Functions

- void [fasp_smoother_dbsr_jacobi](#) ([dBSRmat](#) *A, [dvector](#) *b, [dvector](#) *u)

Jacobi relaxation.
- void [fasp_smoother_dbsr_jacobi_setup](#) ([dBSRmat](#) *A, [dvector](#) *b, [dvector](#) *u, [REAL](#) *diaginv)

Setup for jacobi relaxation, fetch the diagonal sub-block matrixes and make them inverse first.
- void [fasp_smoother_dbsr_jacobi1](#) ([dBSRmat](#) *A, [dvector](#) *b, [dvector](#) *u, [REAL](#) *diaginv)

Jacobi relaxation.
- void [fasp_smoother_dbsr_gs](#) ([dBSRmat](#) *A, [dvector](#) *b, [dvector](#) *u, [INT](#) order, [INT](#) *mark)

Gauss-Seidel relaxation.
- void [fasp_smoother_dbsr_gs1](#) ([dBSRmat](#) *A, [dvector](#) *b, [dvector](#) *u, [INT](#) order, [INT](#) *mark, [REAL](#) *diaginv)

Gauss-Seidel relaxation.
- void [fasp_smoother_dbsr_gs_ascend](#) ([dBSRmat](#) *A, [dvector](#) *b, [dvector](#) *u, [REAL](#) *diaginv)

Gauss-Seidel relaxation in the ascending order.
- void [fasp_smoother_dbsr_gs_ascend1](#) ([dBSRmat](#) *A, [dvector](#) *b, [dvector](#) *u)

Gauss-Seidel relaxation in the ascending order.
- void [fasp_smoother_dbsr_gs_descend](#) ([dBSRmat](#) *A, [dvector](#) *b, [dvector](#) *u, [REAL](#) *diaginv)

Gauss-Seidel relaxation in the descending order.
- void [fasp_smoother_dbsr_gs_descend1](#) ([dBSRmat](#) *A, [dvector](#) *b, [dvector](#) *u)

Gauss-Seidel relaxation in the descending order.
- void [fasp_smoother_dbsr_gs_order1](#) ([dBSRmat](#) *A, [dvector](#) *b, [dvector](#) *u, [REAL](#) *diaginv, [INT](#) *mark)

Gauss-Seidel relaxation in the user-defined order.
- void [fasp_smoother_dbsr_gs_order2](#) ([dBSRmat](#) *A, [dvector](#) *b, [dvector](#) *u, [INT](#) *mark, [REAL](#) *work)

Gauss-Seidel relaxation in the user-defined order.
- void [fasp_smoother_dbsr_sor](#) ([dBSRmat](#) *A, [dvector](#) *b, [dvector](#) *u, [INT](#) order, [INT](#) *mark, [REAL](#) weight)

SOR relaxation.

- void `fasp_smoother_dbsr_sor1` (`dBSRmat *A`, `dvector *b`, `dvector *u`, `INT order`, `INT *mark`, `REAL *diaginv`, `REAL weight`)
SOR relaxation.
- void `fasp_smoother_dbsr_sor_ascend` (`dBSRmat *A`, `dvector *b`, `dvector *u`, `REAL *diaginv`, `REAL weight`)
SOR relaxation in the ascending order.
- void `fasp_smoother_dbsr_sor_descend` (`dBSRmat *A`, `dvector *b`, `dvector *u`, `REAL *diaginv`, `REAL weight`)
SOR relaxation in the descending order.
- void `fasp_smoother_dbsr_sor_order` (`dBSRmat *A`, `dvector *b`, `dvector *u`, `REAL *diaginv`, `INT *mark`, `REAL weight`)
SOR relaxation in the user-defined order.
- void `fasp_smoother_dbsr_ilu` (`dBSRmat *A`, `dvector *b`, `dvector *x`, void `*data`)
ILU method as the smoother in solving $Au=b$ with multigrid method.

9.86.1 Detailed Description

Smoothers for `dBSRmat` matrices.

Definition in file `smoother_bsr.c`.

9.86.2 Function Documentation

9.86.2.1 void fasp_smoother_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 <code>mark = NULL</code> ASCEND 12: in ascending order DESCEND 21: in descending order If <code>mark != NULL</code> : 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`.

9.86.2.2 void fasp_smoother_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 DESCEND 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.

9.86.2.3 void fasp_smoother_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.

9.86.2.4 void fasp_smoother_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_smoother_dbsr_gs_ascend1' and 'fasp_smoother_dbsr_gs_ascend' is that we don't have to multiply by the inverses of the diagonal blocks in each ROW since matrix A has been such scaled that all the diagonal blocks become identity matrices.

Definition at line 645 of file smoother_bsr.c.

9.86.2.5 void fasp_smoother_dbsr_gs_descend (dBSRmat * A, dvector * b, dvector * u, REAL * diaginvs)

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 A

Author

Zhiyang Zhou

Date

2010/10/25

Definition at line 716 of file smoother_bsr.c.

9.86.2.6 void fasp_smoother_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_smoother_dbsr_gs_ascend1' and 'fasp_smoother_dbsr_gs_ascend' is that we don't have to multiply by the inverses of the diagonal blocks in each ROW since matrix A has been such scaled that all the diagonal blocks become identity matrices.

Definition at line 790 of file smoother_bsr.c.

9.86.2.7 void fasp_smoother_dbsr_gs_order1 (dBSRmat * A, dvector * b, dvector * u, REAL * diagin, 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.

9.86.2.8 void fasp_smoother_dbsr_gs_order2 (dBSRmat * A, dvector * b, dvector * u, INT * mark, REAL * work)

Gauss-Seidel relaxation in the user-defined order.

Parameters

<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_smoother_dbsr_gs_order2' and 'fasp_smoother_dbsr_gs_order1' lies in that we don't have to multiply by the inverses of the diagonal blocks in each ROW since matrix A has been such scaled that all the diagonal blocks become identity matrices.

Definition at line 940 of file smoother_bsr.c.

9.86.2.9 void fasp_smoother_dbsr_ilu (dBSRmat * *A*, dvector * *b*, dvector * *x*, void * *data*)

ILU method as the smoother in solving $Au=b$ with multigrid method.

Parameters

<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 $zr = b - A x$

solve LU $z=zr$

$x=x+z$

Definition at line 1573 of file smoother_bsr.c.

9.86.2.10 void fasp_smoother_dbsr_jacobi (dBSRmat * *A*, dvector * *b*, dvector * *u*)

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)

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.

9.86.2.11 void fasp_smoother_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 A

Author

Zhiyang Zhou

Date

2010/10/25

Modified by Chunsheng Feng, Zheng Li on 08/03/2012

Definition at line 259 of file smoother_bsr.c.

9.86.2.12 void fasp_smoother_dbsr_jacobi_setup (dBSRmat * *A*, dvector * *b*, dvector * *u*, REAL * *diaginv*)

Setup for jacobi relaxation, fetch the diagonal sub-block matrixes and make them inverse first.

Parameters

<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.

9.86.2.13 void fasp_smoother_dbsr_sor (dBSRmat * *A*, dvector * *b*, dvector * *u*, INT *order*, INT * *mark*, 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 DESCEND 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>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.

9.86.2.14 void fasp_smoother_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 DESCEND 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
<i>weight</i>	Over-relaxation weight

Author

Zhiyang Zhou

Date

2010/10/25

Definition at line 1141 of file smoother_bsr.c.

9.86.2.15 void fasp_smoother_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 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>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.

9.86.2.16 void fasp_smoother_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 A
<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.

9.86.2.17 void fasp_smoother_dbsr_sor_order (dBSRmat * *A*, dvector * *b*, dvector * *u*, REAL * *diaginv*, INT * *mark*, REAL *weight*)

SOR relaxation in the user-defined order.

Parameters

<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>diagin</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.

9.87 smoother_csr.c File Reference

Smoothers for [dCSRmat](#) matrices.

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

Functions

- void [fasp_smoother_dcsr_jacobi](#) (dvector *u, const INT i_1, const INT i_n, const INT s, [dCSRmat](#) *A, dvector *b, INT L)
Jacobi method as a smoother.
- void [fasp_smoother_dcsr_gs](#) (dvector *u, const INT i_1, const INT i_n, const INT s, [dCSRmat](#) *A, dvector *b, INT L)
Gauss-Seidel method as a smoother.
- void [fasp_smoother_dcsr_gs_cf](#) (dvector *u, [dCSRmat](#) *A, dvector *b, INT L, INT *mark, const INT order)
Gauss-Seidel smoother with C/F ordering for Au=b.
- void [fasp_smoother_dcsr_sgs](#) (dvector *u, [dCSRmat](#) *A, dvector *b, INT L)
Symmetric Gauss-Seidel method as a smoother.
- void [fasp_smoother_dcsr_sor](#) (dvector *u, const INT i_1, const INT i_n, const INT s, [dCSRmat](#) *A, dvector *b, INT L, const REAL w)
SOR method as a smoother.
- void [fasp_smoother_dcsr_sor_cf](#) (dvector *u, [dCSRmat](#) *A, dvector *b, INT L, const REAL w, INT *mark, const INT order)
SOR smoother with C/F ordering for Au=b.
- void [fasp_smoother_dcsr_ilu](#) ([dCSRmat](#) *A, dvector *b, dvector *x, void *data)
ILU method as a smoother.

- void `fasp_smoother_dcsr_kaczmarz` (`dvector` *u, const `INT` i_1, const `INT` i_n, const `INT` s, `dCSRmat` *A, `dvector` *b, `INT` L, const `REAL` w)
Kaczmarz method as a smoother.
- void `fasp_smoother_dcsr_L1diag` (`dvector` *u, const `INT` i_1, const `INT` i_n, const `INT` s, `dCSRmat` *A, `dvector` *b, `INT` L)
Diagonal scaling (using L1 norm) as a smoother.
- void `fasp_smoother_dcsr_gs_rb3d` (`dvector` *u, `dCSRmat` *A, `dvector` *b, `INT` L, `INT` order, `INT` *mark, `INT` maximap, `INT` nx, `INT` ny, `INT` nz)

9.87.1 Detailed Description

Smoothers for `dCSRmat` matrices.

Definition in file `smoother_csr.c`.

9.87.2 Function Documentation

9.87.2.1 void `fasp_smoother_dcsr_gs` (`dvector` * u, const `INT` i_1, const `INT` i_n, const `INT` s, `dCSRmat` * A, `dvector` * b, `INT` L)

Gauss-Seidel method as a smoother.

Parameters

<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>dBSRmat</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`.

9.87.2.2 void `fasp_smoother_dcsr_gs_cf` (`dvector` * u, `dCSRmat` * A, `dvector` * b, `INT` L, `INT` * mark, const `INT` order)

Gauss-Seidel smoother with C/F ordering for $Au=b$.

Parameters

<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.

9.87.2.3 void fasp_smoother_dcsr_ilu (dCSRmat * *A*, dvector * *b*, dvector * *x*, void * *data*)

ILU method as a smoother.

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

Shiquan Zhang, Xiaozhe Hu

Date

2010/11/12

form residual $zr = b - A x$

Definition at line 1067 of file smoother_csr.c.

9.87.2.4 void fasp_smoother_dcsr_jacobi (dvector * *u*, const INT *i_1*, const INT *i_n*, const INT *s*, dCSRmat * *A*, dvector * *b*, INT *L*)

Jacobi method as a smoother.

Parameters

u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
i_1	Starting index
i_n	Ending index
s	Increasing step
A	Pointer to dBSRmat : the coefficient matrix
b	Pointer to dvector: the right hand side
L	Number of iterations

Author

Xuehai Huang, Chensong Zhang

Date

09/26/2009

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

Definition at line 59 of file smoother_csr.c.

9.87.2.5 void fasp_smoother_dcsr_kaczmarz (dvector * u , const INT i_1 , const INT i_n , const INT s , dCSRmat * A , dvector * b , INT L , const REAL w)

Kaczmarz method as a smoother.

Parameters

u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
i_1	Starting index
i_n	Ending index
s	Increasing step
A	Pointer to dBSRmat : the coefficient matrix
b	Pointer to dvector: the right hand side
L	Number of iterations
w	Over-relaxation weight

Author

Xiaozhe Hu

Date

2010/11/12

Modified by Chunsheng Feng, Zheng Li on 2012/09/01

Definition at line 1144 of file smoother_csr.c.

9.87.2.6 void fasp_smoother_dcsr_L1diag (dvector * u , const INT i_1 , const INT i_n , const INT s , dCSRmat * A , dvector * b , INT L)

Diagonal scaling (using L1 norm) as a smoother.

Parameters

u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
i_1	Starting index
i_n	Ending index
s	Increasing step
A	Pointer to dBSRmat : the coefficient matrix
b	Pointer to dvector: the right hand side
L	Number of iterations

Author

Xiaozhe Hu, James Brannick

Date

01/26/2011

Modified by Chunsheng Feng, Zheng Li on 09/01/2012

Definition at line 1285 of file smoother_csr.c.

9.87.2.7 void fasp_smoother_dcsr_sgs (dvector * u , dCSRmat * A , dvector * b , INT L)

Symmetric Gauss-Seidel method as a smoother.

Parameters

u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
A	Pointer to dBSRmat : the coefficient matrix
b	Pointer to dvector: the right hand side
L	Number of iterations

Author

Xiaozhe Hu

Date

10/26/2010

Modified by Chunsheng Feng, Zheng Li on 09/01/2012

Definition at line 629 of file smoother_csr.c.

9.87.2.8 void fasp_smoother_dcsr_sor (dvector * u , const INT i_1 , const INT i_n , const INT s , dCSRmat * A , dvector * b , INT L , const REAL w)

SOR method as a smoother.

Parameters

<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
<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.

9.87.2.9 void fasp_smoother_dcsr_sor_cf (dvector * *u*, dCSRmat * *A*, dvector * *b*, INT *L*, const REAL *w*, INT * *mark*, const INT *order*)

SOR smoother with C/F ordering for $Au=b$.

Parameters

<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>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.

9.88 smoother_csr_cr.c File Reference

Smoother for [dCSRmat](#) matrices using compatible relaxation.

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

Functions

- void [fasp_smoother_dcsr_gscr](#) (INT pt, INT n, REAL *u, INT *ia, INT *ja, REAL *a, REAL *b, INT L, INT *CF)
Gauss Seidel method restriced to a block.

9.88.1 Detailed Description

Smoothers for [dCSRmat](#) matrices using compatible relaxation.

Note

Restricted-smoothers for compatible relaxation, C/F smoothing, etc.

Definition in file [smoother_csr_cr.c](#).

9.88.2 Function Documentation

9.88.2.1 void [fasp_smoother_dcsr_gscr](#) (INT pt, INT n, REAL * u, INT * ia, INT * ja, REAL * a, REAL * b, INT L, INT * CF)

Gauss Seidel method restriced to a block.

Parameters

<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](#).

9.89 smoother_csr_poly.c File Reference

Smoothers for [dCSRmat](#) matrices using poly. approx. to A^{-1} .

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

Functions

- void [fasp_smoother_dcsr_poly](#) ([dCSRmat](#) *Amat, [dvector](#) *brhs, [dvector](#) *usol, [INT](#) n, [INT](#) ndeg, [INT](#) L)
poly approx to A^{-1} as MG smoother
- void [fasp_smoother_dcsr_poly_old](#) ([dCSRmat](#) *Amat, [dvector](#) *brhs, [dvector](#) *usol, [INT](#) n, [INT](#) ndeg, [INT](#) L)
poly approx to A^{-1} as MG smoother: JK<Z2010

9.89.1 Detailed Description

Smoothers for [dCSRmat](#) matrices using poly. approx. to A^{-1} .

Definition in file [smoother_csr_poly.c](#).

9.89.2 Function Documentation

9.89.2.1 void [fasp_smoother_dcsr_poly](#) ([dCSRmat](#) * Amat, [dvector](#) * brhs, [dvector](#) * usol, [INT](#) n, [INT](#) ndeg, [INT](#) L)

poly approx to A^{-1} as MG smoother

Parameters

<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 46 of file [smoother_csr_poly.c](#).

9.89.2.2 void [fasp_smoother_dcsr_poly_old](#) ([dCSRmat](#) * Amat, [dvector](#) * brhs, [dvector](#) * usol, [INT](#) n, [INT](#) ndeg, [INT](#) L)

poly approx to A^{-1} as MG smoother: JK<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 141 of file smoother_csr_poly.c.

9.90 smoother_str.c File Reference

Smoothers for [dSTRmat](#) matrices.

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

Functions

- void [fasp_smoother_dstr_jacobi](#) ([dSTRmat](#) *A, [dvector](#) *b, [dvector](#) *u)
Jacobi method as the smoother.
- void [fasp_smoother_dstr_jacobi1](#) ([dSTRmat](#) *A, [dvector](#) *b, [dvector](#) *u, [REAL](#) *diaginv)
Jacobi method as the smoother with diag_inv given.
- void [fasp_smoother_dstr_gs](#) ([dSTRmat](#) *A, [dvector](#) *b, [dvector](#) *u, [INT](#) order, [INT](#) *mark)
Gauss-Seidel method as the smoother.
- void [fasp_smoother_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_smoother_dstr_gs_ascend](#) ([dSTRmat](#) *A, [dvector](#) *b, [dvector](#) *u, [REAL](#) *diaginv)
Gauss-Seidel method as the smoother in the ascending manner.
- void [fasp_smoother_dstr_gs_descend](#) ([dSTRmat](#) *A, [dvector](#) *b, [dvector](#) *u, [REAL](#) *diaginv)
Gauss-Seidel method as the smoother in the descending manner.
- void [fasp_smoother_dstr_gs_order](#) ([dSTRmat](#) *A, [dvector](#) *b, [dvector](#) *u, [REAL](#) *diaginv, [INT](#) *mark)
Gauss method as the smoother in the user-defined order.
- void [fasp_smoother_dstr_gs_cf](#) ([dSTRmat](#) *A, [dvector](#) *b, [dvector](#) *u, [REAL](#) *diaginv, [INT](#) *mark, [INT](#) order)
Gauss method as the smoother in the C-F manner.
- void [fasp_smoother_dstr_sor](#) ([dSTRmat](#) *A, [dvector](#) *b, [dvector](#) *u, [INT](#) order, [INT](#) *mark, [REAL](#) weight)
SOR method as the smoother.

- void `fasp_smoother_dstr_sor1` (`dSTRmat *A`, `dvector *b`, `dvector *u`, `INT order`, `INT *mark`, `REAL *diaginv`, `REAL weight`)
SOR method as the smoother.
- void `fasp_smoother_dstr_sor_ascend` (`dSTRmat *A`, `dvector *b`, `dvector *u`, `REAL *diaginv`, `REAL weight`)
SOR method as the smoother in the ascending manner.
- void `fasp_smoother_dstr_sor_descend` (`dSTRmat *A`, `dvector *b`, `dvector *u`, `REAL *diaginv`, `REAL weight`)
SOR method as the smoother in the descending manner.
- void `fasp_smoother_dstr_sor_order` (`dSTRmat *A`, `dvector *b`, `dvector *u`, `REAL *diaginv`, `INT *mark`, `REAL weight`)
SOR method as the smoother in the user-defined order.
- void `fasp_smoother_dstr_sor_cf` (`dSTRmat *A`, `dvector *b`, `dvector *u`, `REAL *diaginv`, `INT *mark`, `INT order`, `REAL weight`)
SOR method as the smoother in the C-F manner.
- void `fasp_generate_diaginv_block` (`dSTRmat *A`, `ivector *neigh`, `dvector *diaginv`, `ivector *pivot`)
Generate inverse of diagonal block for block smoothers.
- void `fasp_smoother_dstr_schwarz` (`dSTRmat *A`, `dvector *b`, `dvector *u`, `dvector *diaginv`, `ivector *pivot`, `ivector *neigh`, `ivector *order`)
Schwarz method as the smoother.

9.90.1 Detailed Description

Smoothers for `dSTRmat` matrices.

Definition in file `smoother_str.c`.

9.90.2 Function Documentation

9.90.2.1 void fasp_generate_diaginv_block (dSTRmat * A, ivector * neigh, dvector * diaginv, ivector * pivot)

Generate inverse of diagonal block for block smoothers.

Parameters

<i>A</i>	Pointer to <code>dCSRmat</code> : the coefficient matrix
<i>neigh</i>	Pointer to <code>ivector</code> : neighborhoods
<i>diaginv</i>	Pointer to <code>dvector</code> : the inverse of the diagonals
<i>pivot</i>	Pointer to <code>ivector</code> : the pivot of diagonal blocks

Author

Xiaozhe Hu

Date

10/01/2011

Definition at line 1517 of file `smoother_str.c`.

9.90.2.2 void fasp_smoother_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-DESCEND 21: in descending manner If mark != NULL USERDEFINED 0 : in the user-defined manner CPFIRST 1 : C-points first and then F-points FPFIRST -1 : F-points first and then C-points
<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.

9.90.2.3 void fasp_smoother_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-DESCEND 21: in descending manner If mark != NULL USERDEFINED 0 : in the user-defined manner CPFIRST 1 : C-points first and then F-points FPFIRST -1 : F-points first and then C-points
<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.

9.90.2.4 void fasp_smoother_dstr_gs_ascend (dSTRmat * *A*, dvector * *b*, dvector * *u*, REAL * *diaginv*)

Gauss-Seidel method as the smoother in the ascending manner.

Parameters

<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 306 of file smoother_str.c.

9.90.2.5 void fasp_smoother_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 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

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 660 of file smoother_str.c.

9.90.2.6 void fasp_smoother_dstr_gs_descend (dSTRmat * *A*, dvector * *b*, dvector * *u*, REAL * *diaginv*)

Gauss-Seidel method as the smoother in the descending manner.

Parameters

<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.

9.90.2.7 void fasp_smoother_dstr_gs_order (dSTRmat * *A*, dvector * *b*, dvector * *u*, REAL * *diaginv*, INT * *mark*)

Gauss method as the smoother in the user-defined order.

Parameters

<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.

9.90.2.8 void fasp_smoother_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.

9.90.2.9 void fasp_smoother_dstr_jacobi1 (dSTRmat * *A*, dvector * *b*, dvector * *u*, REAL * *diaginv*)

Jacobi 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>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 79 of file smoother_str.c.

9.90.2.10 void fasp_smoother_dstr_schwarz (dSTRmat * *A*, dvector * *b*, dvector * *u*, dvector * *diaginv*, ivector * *pivot*, ivector * *neigh*, ivector * *order*)

Schwarz method as the smoother.

Parameters

<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 ivector: the pivot of diagonal blocks
<i>neigh</i>	Pointer to ivector: neighborhoods
<i>order</i>	Pointer to ivector: the smoothing order

Author

Xiaozhe Hu

Date

10/01/2011

Definition at line 1639 of file smoother_str.c.

9.90.2.11 void fasp_smoother_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-DESCEND 21: in descending manner If mark != NULL USERDEFINED 0 : in the user-defined manner CPFIRST 1 : C-points first and then F-points FPFIRST -1 : F-points first and then C-points
<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.

9.90.2.12 void fasp_smoother_dstr_sor1 (dSTRmat * *A*, dvector * *b*, dvector * *u*, INT *order*, INT * *mark*, REAL * *diaginv*, 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-DESCEND 21: in descending manner If mark != NULL USERDEFINED 0 : in the user-defined manner CPFIRST 1 : C-points first and then F-points FPFIRST -1 : F-points first and then C-points
<i>mark</i>	Pointer to the user-defined ordering(when order=0) or CF_marker array(when order!=0)
<i>diaginv</i>	Inverse of the diagonal entries
<i>weight</i>	Over-relaxation weight

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 912 of file smoother_str.c.

9.90.2.13 void fasp_smoother_dstr_sor_ascend (dSTRmat * *A*, dvector * *b*, dvector * *u*, REAL * *diaginv*, REAL *weight*)

SOR method as the smoother in the ascending manner.

Parameters

<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>weight</i>	Over-relaxation weight

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 958 of file smoother_str.c.

9.90.2.14 void fasp_smoother_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.

9.90.2.15 void fasp_smoother_dstr_sor_descend (dSTRmat * *A*, dvector * *b*, dvector * *u*, REAL * *diaginv*, REAL *weight*)

SOR method as the smoother in the descending manner.

Parameters

<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>weight</i>	Over-relaxation weight

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 1078 of file smoother_str.c.

9.90.2.16 void fasp_smoother_dstr_sor_order (dSTRmat * *A*, dvector * *b*, dvector * *u*, REAL * *diaginv*, INT * *mark*, REAL *weight*)

SOR method as the smoother in the user-defined order.

Parameters

<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>weight</i>	Over-relaxation weight

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 1199 of file smoother_str.c.

9.91 sparse_block.c File Reference

Sparse matrix block operations.

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

Functions

- [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

9.91.1 Detailed Description

Sparse matrix block operations.

Definition in file [sparse_block.c](#).

9.91.2 Function Documentation

9.91.2.1 [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 dBSRmat BSR matrix
<i>B</i>	Pointer to dBSRmat 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

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 132 of file [sparse_block.c](#).

9.91.2.2 [dCSRmat fasp_dbsr_getblk_dcsr](#) ([dBSRmat](#) * A)

get [dCSRmat](#) block from a [dBSRmat](#) matrix

Parameters

<i>*A</i>	Pointer to the BSR format matrix
-----------	----------------------------------

Returns

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

Author

Xiaozhe Hu

Date

03/16/2012

Definition at line 226 of file sparse_block.c.

9.91.2.3 `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.

Parameters

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

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 39 of file sparse_block.c.

9.92 `sparse_bsr.c` File Reference

Sparse matrix operations for [dBSRmat](#) matrices.

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


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, dBSRmat *A)
Allocate memory space for BSR format sparse matrix.
- void [fasp_dbsr_free](#) (dBSRmat *A)
Free memeory 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_getdiainv](#) (dBSRmat *A)
Get D^{-1} of matrix A.
- dBSRmat [fasp_dbsr_diainv](#) (dBSRmat *A)
*Compute $B := D^{-1} * A$, where 'D' is the block diagonal part of A.*
- dBSRmat [fasp_dbsr_diainv2](#) (dBSRmat *A, REAL *diainv)
*Compute $B := D^{-1} * A$, where 'D' is the block diagonal part of A.*
- dBSRmat [fasp_dbsr_diainv3](#) (dBSRmat *A, REAL *diainv)
*Compute $B := D^{-1} * A$, where 'D' is the block diagonal part of A.*
- dBSRmat [fasp_dbsr_diainv4](#) (dBSRmat *A, REAL *diainv)
*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.

9.92.1 Detailed Description

Sparse matrix operations for dBSRmat matrices.

Definition in file [sparse_bsr.c](#).

9.92.2 Function Documentation

9.92.2.1 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

ROW	Number of rows of block
COL	Number of columns of block

<i>NNZ</i>	Number of nonzero blocks
<i>nb</i>	Dimension of exch 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 84 of file sparse_bsr.c.

9.92.2.2 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 178 of file sparse_bsr.c.

9.92.2.3 [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 exch 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 33 of file sparse_bsr.c.

9.92.2.4 dBSRmat fasp_dbsr_diaginv (dBSRmat * A)Compute $B := D^{-1} * A$, 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 502 of file sparse_bsr.c.

9.92.2.5 dBSRmat fasp_dbsr_diaginv2 (dBSRmat * A, REAL * diaginv)Compute $B := D^{-1} * A$, 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

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 664 of file sparse_bsr.c.

9.92.2.6 dBSRmat fasp_dbsr_diaginv3 (dBSRmat * A, REAL * diaginv)Compute $B := D^{-1} * A$, 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

Author

Xiaozhe Hu

Date

12/25/2010

Note

Works for general nb (Xiaozhe)

Modified by Xiaozhe Hu on 05/26/2012

Definition at line 766 of file sparse_bsr.c.

9.92.2.7 [dBSRmat](#) fasp_dbsr_diaginv4 ([dBSRmat](#) * *A*, **REAL** * *diaginv*)

Compute $B := D^{-1} * A$, 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 preordered 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 1124 of file sparse_bsr.c.

9.92.2.8 **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 289 of file sparse_bsr.c.

9.92.2.9 void fasp_dbsr_free (dBSRmat * *A*)

Free memory space for BSR format sparse matrix.

Parameters

<i>A</i>	Pointer to the dBSRmat matrix
----------	---

Author

Xiaozhe Hu

Date

10/26/2010

Definition at line 130 of file sparse_bsr.c.

9.92.2.10 fasp_dbsr_getdiag (INT *n*, dBSRmat * *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 dBSRmat 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 1417 of file sparse_bsr.c.

9.92.2.11 dvector fasp_dbsr_getdiagin (dBSRmat * A)Get D^{-1} of matrix A.

Parameters

<i>A</i>	Pointer to the dBSRmat matrix
----------	---

Author

Xiaozhe Hu

Date

02/19/2013

Note

Works for general nb (Xiaozhe)

Definition at line 398 of file sparse_bsr.c.

9.92.2.12 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 155 of file sparse_bsr.c.

9.92.2.13 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 205 of file sparse_bsr.c.

9.93 sparse_coo.c File ReferenceSparse matrix operations for [dCOOmat](#) matrices.

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

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 memeory space.
- void [fasp_dcoo_shift](#) ([dCOOmat](#) *A, INT offset)
Reindex a REAL matrix in IJ format to make the index starting from 0 or 1.

9.93.1 Detailed Description

Sparse matrix operations for [dCOOmat](#) matrices.

Definition in file [sparse_coo.c](#).

9.93.2 Function Documentation

9.93.2.1 `void fasp_dcoo_alloc (const INT m, const INT n, const INT nnz, dCOOmat * A)`

Allocate COO 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

Xiaozhe Hu

Date

03/25/2013

Definition at line 62 of file [sparse_coo.c](#).

9.93.2.2 `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](#).

9.93.2.3 `void fasp_dcoo_free (dCOOmat * A)`

Free IJ sparse matrix data memeory space.

Parameters

<i>A</i>	Pointer to the dCOOmat matrix
----------	---

Author

Chensong Zhang

Date

2010/04/03

Definition at line 95 of file sparse_coo.c.

9.93.2.4 void fasp_dcoo_shift ([dCOOmat](#) * *A*, *INT offset*)

Reindex 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 117 of file sparse_coo.c.

9.94 sparse_csr.c File Reference

Sparse matrix operations for [dCSRmat](#) matrices.

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

Functions

- [dCSRmat fasp_dcsr_create](#) (const [INT](#) m, const [INT](#) n, const [INT](#) nnz)
Create CSR sparse matrix data memory space.
- [iCSRmat fasp_icsr_create](#) (const [INT](#) m, const [INT](#) n, const [INT](#) nnz)
Create CSR sparse matrix data memory space.
- void [fasp_dcsr_alloc](#) (const [INT](#) m, const [INT](#) n, const [INT](#) nnz, [dCSRmat](#) *A)

- *Allocate CSR sparse matrix memory space.*
- void [fasp_dcsr_free](#) ([dCSRmat](#) *A)
- *Free CSR sparse matrix data memeory space.*
- void [fasp_icsr_free](#) ([iCSRmat](#) *A)
- *Free CSR sparse matrix data memeory space.*
- void [fasp_dcsr_null](#) ([dCSRmat](#) *A)
- *Initialize CSR sparse matrix.*
- void [fasp_icsr_null](#) ([iCSRmat](#) *A)
- *Initialize CSR sparse matrix.*
- [dCSRmat](#) [fasp_dcsr_perm](#) ([dCSRmat](#) *A, [INT](#) *P)
- *Apply permutation of A, i.e. Aperm=PAP' by the orders given in P.*
- void [fasp_dcsr_sort](#) ([dCSRmat](#) *A)
- *Sort each row of A in ascending order w.r.t. column indices.*
- void [fasp_dcsr_getdiag](#) ([INT](#) n, [dCSRmat](#) *A, [dvector](#) *diag)
- *Get first n diagonal entries of a CSR matrix A.*
- void [fasp_dcsr_getcol](#) (const [INT](#) n, [dCSRmat](#) *A, [REAL](#) *col)
- *Get the n-th column of a CSR matrix A.*
- void [fasp_dcsr_diagpref](#) ([dCSRmat](#) *A)
- *Re-order the column and data arrays of a CSR matrix, so that the first entry in each row is the diagonal.*
- [SHORT](#) [fasp_dcsr_regdiag](#) ([dCSRmat](#) *A, [REAL](#) value)
- *Regularize diagonal entries of a CSR sparse matrix.*
- void [fasp_icsr_cp](#) ([iCSRmat](#) *A, [iCSRmat](#) *B)
- *Copy a iCSRmat to a new one B=A.*
- void [fasp_dcsr_cp](#) ([dCSRmat](#) *A, [dCSRmat](#) *B)
- *copy a dCSRmat to a new one B=A*
- void [fasp_icsr_trans](#) ([iCSRmat](#) *A, [iCSRmat](#) *AT)
- *Find transpose of iCSRmat matrix A.*
- [INT](#) [fasp_dcsr_trans](#) ([dCSRmat](#) *A, [dCSRmat](#) *AT)
- *Find tranpose of dCSRmat matrix A.*
- void [fasp_dcsr_transpose](#) ([INT](#) *row[2], [INT](#) *col[2], [REAL](#) *val[2], [INT](#) *nn, [INT](#) *tniz)
- void [fasp_dcsr_compress](#) ([dCSRmat](#) *A, [dCSRmat](#) *B, [REAL](#) dtol)
- *Compress a CSR matrix A and store in CSR matrix B by dropping small entries $abs(a_{ij}) \leq dtol$.*
- [SHORT](#) [fasp_dcsr_compress_inplace](#) ([dCSRmat](#) *A, [REAL](#) dtol)
- *Compress a CSR matrix A IN PLACE by dropping small entries $abs(a_{ij}) \leq dtol$.*
- void [fasp_dcsr_shift](#) ([dCSRmat](#) *A, [INT](#) offset)
- *Reindex 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)
- *Symmetrize the parttarn of a dCSRmat matrix.*
- void [fasp_dcsr_multicoloring](#) ([dCSRmat](#) *A, [INT](#) *flags, [INT](#) *groups)
- *Use the greedy multicoloring to get color groups of the adjacency graph of A.*

9.94.1 Detailed Description

Sparse matrix operations for [dCSRmat](#) matrices.

Definition in file [sparse_csr.c](#).

9.94.2 Function Documentation

9.94.2.1 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.

9.94.2.2 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

<i>A</i>	Pointer to dCSRmat CSR matrix
<i>B</i>	Pointer to dCSRmat CSR matrix
<i>dtol</i>	Drop tolerance

Author

Shiquan Zhang

Date

03/10/2010

Modified by Chunsheng Feng, Zheng Li on 08/25/2012

Definition at line 960 of file sparse_csr.c.

9.94.2.3 SHORT fasp_dcsr_compress_inplace (dCSRmat * *A*, REAL *dtol*)

Compress a CSR matrix A IN PLACE by dropping small entries $\text{abs}(a_{ij}) \leq \text{dtol}$.

Parameters

<i>A</i>	Pointer to dCSRmat CSR matrix
<i>dtol</i>	Drop tolerance

Author

Xiaozhe Hu

Date

12/25/2010

Modified by Chensong on 02/21/2013

Note

This routine can be modified for filtering.

Definition at line 1040 of file sparse_csr.c.

9.94.2.4 `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 722 of file sparse_csr.c.

9.94.2.5 `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.

9.94.2.6 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.

9.94.2.7 void fasp_dcsr_free (dCSRmat * A)

Free CSR sparse matrix data memeory space.

Parameters

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

Author

Chensong Zhang

Date

2010/04/06

Definition at line 166 of file sparse_csr.c.

9.94.2.8 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.

9.94.2.9 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 diag 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.

9.94.2.10 void fasp_dcsr_multicoloring (dCSRmat * *A*, INT * *flags*, INT * *groups*)

Use the greedy multicoloring 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 1272 of file sparse_csr.c.

9.94.2.11 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.

9.94.2.12 [dCSRmat](#) fasp_dcsr_perm ([dCSRmat](#) * *A*, INT * *P*)

Apply permutation of A, i.e. Aperm=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.

9.94.2.13 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(A) which is too close to zero to "value"

Returns

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

Author

Shiquan Zhang

Date

11/07/2009

Definition at line 658 of file sparse_csr.c.

9.94.2.14 void fasp_dcsr_shift (dCSRmat * A, INT offset)

Reindex 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 1088 of file sparse_csr.c.

9.94.2.15 void fasp_dcsr_sort (dCSRmat * A)

Sort each row of A in ascending order w.r.t. column indices.

Parameters

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

Author

Shiquan Zhang

Date

06/10/2010

Definition at line 356 of file sparse_csr.c.

9.94.2.16 void fasp_dcsr_symdiagscale (dCSRmat * *A*, dvector * *diag*)Symmetric diagonal scaling $D^{-1/2}AD^{-1/2}$.

Parameters

<i>A</i>	Pointer to the dCSRmat matrix
<i>diag</i>	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 1151 of file sparse_csr.c.

9.94.2.17 dCSRmat fasp_dcsr_sympat (dCSRmat * *A*)Symmetrize the parttern of a [dCSRmat](#) matrix.

Parameters

* <i>A</i>	pointer to the dCSRmat matrix
------------	---

Returns

symmetrized the [dCSRmat](#) matrix

Author

Xiaozhe Hu

Date

03/21/2011

Definition at line 1238 of file sparse_csr.c.

9.94.2.18 void fasp_dcsr_trans (dCSRmat * A, dCSRmat * AT)

Find tranpose 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 828 of file sparse_csr.c.

9.94.2.19 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 697 of file sparse_csr.c.

9.94.2.20 [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.

9.94.2.21 void fasp_icsr_free (iCSRmat * A)

Free CSR sparse matrix data memeory space.

Parameters

A	Pointer to the iCSRmat matrix
---	---

Author

Chensong Zhang

Date

2010/04/06

Definition at line 185 of file sparse_csr.c.

9.94.2.22 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.

9.94.2.23 void fasp_icsr_trans (iCSRmat * A, iCSRmat * AT)

Find transpose of [iCSRmat](#) matrix A.

Parameters

A	Pointer to the iCSRmat matrix A
---	---

<i>AT</i>	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 749 of file sparse_csr.c.

9.95 sparse_csr.c File Reference

Sparse matrix operations for [dCSRLmat](#) matrices.

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

Functions

- [dCSRLmat](#) * [fasp_dcsr_create](#) ([INT](#) num_rows, [INT](#) num_cols, [INT](#) num_nonzeros)
Create a [dCSRLmat](#) object.
- void [fasp_dcsr_free](#) ([dCSRLmat](#) *A)
Destroy a [dCSRLmat](#) object.

9.95.1 Detailed Description

Sparse matrix operations for [dCSRLmat](#) matrices.

Note

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

Definition in file [sparse_csr.c](#).

9.95.2 Function Documentation

9.95.2.1 [dCSRLmat](#) * [fasp_dcsr_create](#) ([INT](#) num_rows, [INT](#) num_cols, [INT](#) num_nonzeros)

Create a [dCSRLmat](#) 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_csrl.c.

9.95.2.2 void fasp_dcsrl_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_csrl.c.

9.96 sparse_str.c File Reference

Sparse matrix operations for dSTRmat matrices.

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

Functions

- void [fasp_dstr_null](#) (dSTRmat *A)
Initialize sparse matrix on structured grid.
- [dSTRmat fasp_dstr_create](#) (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 memeory space.
- void [fasp_dstr_cp](#) ([dSTRmat](#) *A, [dSTRmat](#) *A1)
Copy a [dSTRmat](#) to a new one A1=A.

9.96.1 Detailed Description

Sparse matrix operations for [dSTRmat](#) matrices.

Definition in file [sparse_str.c](#).

9.96.2 Function Documentation

9.96.2.1 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](#).

9.96.2.2 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.

9.96.2.3 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.

9.96.2.4 void fasp_dstr_free (dSTRmat * *A*)

Free STR sparse matrix data memeory 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.

9.96.2.5 void fasp_dstr_null (dSTRmat * *A*)

Initialize sparse matrix on structured grid.

Parameters

A	Pointer to the dSTRmat matrix
---	---

Author

Shiquan Zhang, Xiaozhe Hu

Date

05/17/2010

Definition at line 25 of file sparse_str.c.

9.97 sparse_util.c File Reference

Routines for sparse matrix operations.

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

Functions

- void [fasp_sparse_abybms_](#) (INT *ia, INT *ja, INT *ib, INT *jb, INT *nap, INT *map, INT *mbp, INT *ic, INT *jc)
Multiplication of two sparse matrices: calculating the nonzero structure of the result if jc is not null. If jc is null only finds num of nonzeros.
- 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 nonzeros.
- 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 $R \cdot A \cdot P$, if jac is not null. If jac is null only finds num of nonzeros.
- 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.
- 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 *npx, INT *icp, REAL *s)

Calculates $s = y^t x$. y is sparse, x is sparse.

- void `fasp_sparse_rapcmp_` (INT *ir, INT *jr, REAL *r, INT *ia, INT *ja, REAL *a, INT *ipt, INT *jpt, REAL *pt, INT *nin, INT *ncin, INT *iac, INT *jac, REAL *ac, INT *idummy)

Calculates $R*A*P$ after the nonzero structure of the result is known. iac,jac,ac have to be allocated before call to this function.

- `ivector fasp_sparse_MIS` (dCSRmat *A)

get the maximal independet set of a CSR matrix

9.97.1 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

Definition in file `sparse_util.c`.

9.97.2 Function Documentation

9.97.2.1 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

<i>ia</i>	array of row pointers (as usual in CSR)
<i>ja</i>	array of column indices
<i>a</i>	array of entries of teh input
<i>na</i>	number of rows of A
<i>ma</i>	number of cols of A
<i>iat</i>	array of row pointers in the result
<i>jat</i>	array of column indices
<i>at</i>	array of entries of the result

Definition at line 272 of file `sparse_util.c`.

9.97.2.2 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

<i>ia</i>	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.

9.97.2.3 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.

9.97.2.4 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`.

9.97.2.5 `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`.

9.97.2.6 `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`.

9.97.2.7 `ivector fasp_sparse_MIS (dCSRmat * A)`

get the maximal independet set of a CSR matrix

Parameters

<i>A</i>	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.

9.97.2.8 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 $R \cdot A \cdot P$ after the nonzero structure of the result is known. *iac,jac,ac* have to be allocated before call to this function.

Note

:I: is input :O: is output :IO: is both

Parameters

<i>ir</i>	:I: array of row pointers for R
<i>jr</i>	:I: array of column indices for R
<i>r</i>	:I: entries of R
<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>ipt</i>	:I: array of row pointers for P
<i>jpt</i>	:I: array of column indices for P
<i>pt</i>	:I: entries of P
<i>nin</i>	:I: number of rows in R
<i>ncin</i>	:I: 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 $R \cdot A \cdot P$ for known nonzero structure of the result the result is stored in *iac,jac,ac*!

Definition at line 791 of file sparse_util.c.

9.97.2.9 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 $R \cdot A \cdot P$, if *jac* is not null. If *jac* is null only finds num of nonzeros.

Note

:I: is input :O: is output :IO: is both

Parameters

<i>ir</i>	:I: array of row pointers for R
<i>jr</i>	:I: array of column indices for R
<i>ia</i>	:I: array of row pointers for A
<i>ja</i>	:I: array of column indices for A
<i>ip</i>	:I: array of row pointers for P
<i>jp</i>	:I: array of column indices for P
<i>nin</i>	:I: number of rows in R
<i>ncin</i>	:I: 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 $R \cdot A \cdot P$. maxrout is output and is the maximum nonzeros per row for r. On output we also have 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.

9.97.2.10 void fasp_sparse_wta_ (INT * *jw*, REAL * *w*, INT * *ia*, INT * *ja*, REAL * *a*, INT * *nwp*, INT * *map*, INT * *jv*, REAL * *v*, INT * *nvp*)

Calculate $v^t = w^t A$, where w is a sparse vector and A is sparse matrix. v is an array of dimension = number of columns in A .

Note

:I: is input :O: is output :IO: is both

Parameters

<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.

9.97.2.11 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 $icp[]$ 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`.

9.97.2.12 `void fasp_sparse_ytx_ (INT * jy, REAL * y, INT * jx, REAL * x, INT * nyp, INT * nxp, INT * icp, REAL * s)`

Calculates $s = y^t x$. y is sparse, x is sparse.

note :I: is input :O: is output :IO: is both

Parameters

<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>	:I: indices such that $x[jx]$ is nonzero
<i>x</i>	:I: is a sparse vector.
<i>nxp</i>	:I: number of nonzeros in x
<i>icp</i>	???
<i>s</i>	:O: $s = y^t x$.

Definition at line 736 of file `sparse_util.c`.

9.97.2.13 `void fasp_sparse_ytxbig_ (INT * jy, REAL * y, INT * nyp, REAL * x, REAL * s)`

Calculates $s = y^t x$. y -sparse, x - no.

Note

: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>x</i>	:I: also a vector assumed to have entry for any $j=jy[i]-1$; for $i=1:nyp$. This means that x here does not have to be sparse.

s	:O: $s = y^t x$.
-----	-------------------

Definition at line 702 of file sparse_util.c.

9.98 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"
```

Functions

- [INT fasp_solver_dcsr_spbcgs](#) (dCSRmat *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 $Au=b$ with safe net.
- [INT fasp_solver_dbsr_spbcgs](#) (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 $Au=b$ with safe net.
- [INT fasp_solver_bdcsr_spbcgs](#) (block_dCSRmat *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 $Au=b$ with safe net.
- [INT fasp_solver_dstr_spbcgs](#) (dSTRmat *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 $Au=b$ with safe net.

9.98.1 Detailed Description

Krylov subspace methods – Preconditioned BiCGstab with safe net. Abstract algorithm

PBICGStab method to solve $A*x=b$ is to generate $\{x_k\}$ to approximate x

Note: We generate a series of $\{p_k\}$ such that $V_k = \text{span}\{p_1, \dots, p_k\}$.

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: $x_{k+1} = x_k + \alpha * p_k$;
- 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 [spbcgs.c](#) for a safer version

Definition in file [spbcgs.c](#).

9.98.2 Function Documentation

9.98.2.1 **INT fasp_solver_bdcsr_spbcgs (block_dCSRmat * 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 $Au=b$ 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 868 of file spbcgs.c.

9.98.2.2 `INT fasp_solver_dbsr_spbcgs (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 $Au=b$ 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 480 of file spbcgs.c.

9.98.2.3 **INT** fasp_solver_dcsr_spgcg (**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.

9.98.2.4 `INT fasp_solver_dstr_spbcgs (dSTRmat * 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 $Au=b$ 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 1256 of file spbcgs.c.

9.99 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"
```

Functions

- `INT fasp_solver_dcsr_spcg (dCSRmat *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 $Au=b$ with safe net.
- `INT fasp_solver_bdcsr_spcg (block_dCSRmat *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 $Au=b$ with safe net.
- `INT fasp_solver_dstr_spcg (dSTRmat *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 $Au=b$ with safe net.

9.99.1 Detailed Description

Krylov subspace methods – Preconditioned conjugate gradient with safe net. Abstract algorithm

PCG method to solve $Ax=b$ is to generate $\{x_k\}$ to approximate x

Step 0. Given A, b, x_0, M

Step 1. Compute residual $r_0 = b - Ax_0$ 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: $r_{k+1} = r_k - \alpha(Ap_k)$;
- 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 $r_{k+1} > r_{\text{best}}$
 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 [pcg.c](#) for a version without safe net

Definition in file [spcg.c](#).

9.99.2 Function Documentation

9.99.2.1 `INT fasp_solver_bdcsr_spcg (block_dCSRmat * 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 $Au=b$ 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

03/28/2013

Definition at line 414 of file spcg.c.

9.99.2.2 `INT fasp_solver_dcsr_spcg (dCSRmat * 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 $Au=b$ 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.

9.99.2.3 `INT fasp_solver_dstr_spcg (dSTRmat * 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 $Au=b$ 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 738 of file spcg.c.

9.100 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"
```

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.

9.100.1 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

Definition in file [spgmres.c](#).

9.100.2 Function Documentation

9.100.2.1 `INT fasp_solver_bdcsr_spgmres (block_dCSRmat * A, dvector * b, dvector * x, precondition * 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 block_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

Definition at line 387 of file [spgmres.c](#).

9.100.2.2 `INT fasp_solver_dbsr_spgmres (dBSRmat * A, dvector * b, dvector * x, precondition * 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 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 728 of file spgmres.c.

9.100.2.3 **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.

9.100.2.4 INT fasp_solver_dstr_spgmres (dSTRmat * A, dvector * b, dvector * x, precondition * 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 1069 of file spgmres.c.

9.101 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"
```

Functions

- [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 $Au=b$ with safe net.
- [INT fasp_solver_bdcsr_spminres](#) (block_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 $Au=b$ with safe net.
- [INT fasp_solver_dstr_spminres](#) (dSTRmat *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 $Au=b$ with safe net.

9.101.1 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 $k = 0:\text{MaxIt}$

- 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 $r_{k+1} < r_{\text{best}}$: 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 $\text{norm}(\alpha*p_k)/\text{norm}(x_{k+1}) < \text{tol_stag}$
 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 $r_{k+1} > r_{\text{best}}$
 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 [pminres.c](#) for a version without safe net

Definition in file [spminres.c](#).

9.101.2 Function Documentation

9.101.2.1 **INT fasp_solver_bdcsr_spminres (block_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 $Au=b$ 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

04/09/2013

Definition at line 543 of file [spminres.c](#).

9.101.2.2 **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 $Au=b$ 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

04/09/2013

Definition at line 96 of file spminres.c.

9.101.2.3 INT fasp_solver_dstr_spminres (dSTRmat * *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 $Au=b$ 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 990 of file spminres.c.

9.102 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"
```

Functions

- **INT fasp_solver_dcsr_spvgmres** (dCSRmat *A, dvector *b, dvector *x, precondition *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, precondition *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, precondition *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, precondition *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.

9.102.1 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 GMRES(m) Journal of Computational and Applied Mathematics, 230 (2009) pp. 751-761. UCRL-JRNL-235266.
See [pvgmres.c](#) a version without safe net

Definition in file [spvgmres.c](#).

9.102.2 Function Documentation

9.102.2.1 INT fasp_solver_bdcsr_spvgmres (block_dCSRmat * A, dvector * b, dvector * x, precondition * 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 block_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

Definition at line 427 of file spvgmres.c.

9.102.2.2 `INT fasp_solver_dbsr_spvgmres (dBSRmat * A, dvector * b, dvector * x, precondition * 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 807 of file spvgmres.c.

9.102.2.3 **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.

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.

9.102.2.4 **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 1187 of file spvgmres.c.

9.103 threads.c File Reference

Get and set number of threads and assigne work load for each thread.

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

Functions

- void [FASP_GET_START_END](#) (INT *procid*, INT *nprocs*, INT *n*, INT **start*, INT **end*)
Assign Load to each thread.

9.103.1 Detailed Description

Get and set number of threads and assigne work load for each thread.

Definition in file [threads.c](#).

9.103.2 Function Documentation

9.103.2.1 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.

9.104 timing.c File Reference

Timing subroutines.

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

Functions

- void [fasp_gettime](#) ([REAL](#) *time)
Get system time.

9.104.1 Detailed Description

Timing subroutines.

Definition in file [timing.c](#).

9.104.2 Function Documentation

9.104.2.1 [fasp_gettime](#) ([REAL](#) * *time*)

Get system time.

Author

Chunsheng Feng, Zheng LI

Date

11/10/2012

Definition at line 28 of file timing.c.

9.105 vec.c File Reference

Simple operations for vectors.

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

Functions

- [INT fasp_dvec_isnan](#) ([dvector](#) *u)
Check a dvector whether there is NAN.
- [dvector fasp_dvec_create](#) (const [INT](#) m)
Create dvector data space of REAL type.
- [ivector fasp_ivec_create](#) (const [INT](#) m)
Create vector data space of INT type.
- void [fasp_dvec_alloc](#) (const [INT](#) m, [dvector](#) *u)
Create dvector data space of REAL type.
- void [fasp_ivec_alloc](#) (const [INT](#) m, [ivector](#) *u)
Create vector data space of INT type.
- void [fasp_dvec_free](#) ([dvector](#) *u)
Free vector data space of REAL type.
- void [fasp_ivec_free](#) ([ivector](#) *u)
Free vector data space of INT type.
- void [fasp_dvec_null](#) ([dvector](#) *x)
Initialize dvector.
- void [fasp_dvec_rand](#) (const [INT](#) n, [dvector](#) *x)
Generate random REAL vector in the range from 0 to 1.
- void [fasp_dvec_set](#) ([INT](#) n, [dvector](#) *x, [REAL](#) val)
Initialize dvector $x[i]=val$ for $i=0:n-1$.
- void [fasp_ivec_set](#) (const [INT](#) m, [ivector](#) *u)
Set ivector value to be m.
- void [fasp_dvec_cp](#) ([dvector](#) *x, [dvector](#) *y)
Copy dvector x to dvector y.
- [REAL fasp_dvec_maxdiff](#) ([dvector](#) *x, [dvector](#) *y)
Maximal difference of two dvector x and y.
- void [fasp_dvec_symdiagscale](#) ([dvector](#) *b, [dvector](#) *diag)
Symmetric diagonal scaling $D^{-1/2}b$.

9.105.1 Detailed Description

Simple operations for vectors.

Note

Every structures should be initialized before usage.

Definition in file [vec.c](#).

9.105.2 Function Documentation

9.105.2.1 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.

9.105.2.2 void fasp_dvec_cp (dvector * *x*, dvector * *y*)

Copy dvector *x* to dvector *y*.

Parameters

<i>x</i>	Pointer to dvector
<i>y</i>	Pointer to dvector (MODIFIED)

Author

Chensong Zhang

Date

11/16/2009

Definition at line 345 of file vec.c.

9.105.2.3 dvector fasp_dvec_create (const INT *m*)

Create dvector data space of REAL type.

Parameters

<i>m</i>	Number of rows
----------	----------------

Returns

u The new dvector

Author

Chensong Zhang

Date

2010/04/06

Definition at line 56 of file vec.c.

9.105.2.4 void fasp_dvec_free (dvector * u)

Free vector data space of REAL type.

Parameters

u	Pointer to dvector which needs to be deallocated
-----	--

Author

Chensong Zhang

Date

2010/04/03

Definition at line 139 of file vec.c.

9.105.2.5 INT fasp_dvec_isnan (dvector * u)

Check a dvector whether there is NAN.

Parameters

u	Pointer to dvector
-----	--------------------

Returns

Return TRUE if there is NAN

Author

Chensong Zhang

Date

2013/03/31

Definition at line 33 of file vec.c.

9.105.2.6 REAL fasp_dvec_maxdiff (dvector * x , dvector * y)Maximal difference of two dvector x and y .

Parameters

x	Pointer to dvector
y	Pointer to dvector

Returns

Maximal norm of $x-y$

Author

Chensong Zhang

Date

11/16/2009

Modified by chunsheng Feng, Zheng Li

Date

06/30/2012

Definition at line 368 of file vec.c.

9.105.2.7 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.

9.105.2.8 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.

9.105.2.9 void fasp_dvec_set (INT n , dvector * x , REAL val)

Initialize dvector $x[i]=val$ for $i=0:n-1$.

Parameters

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

Author

Chensong Zhang

Date

11/16/2009

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

Definition at line 235 of file vec.c.

9.105.2.10 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.

9.105.2.11 void fasp_ivec_alloc (const INT *m*, ivector * *u*)

Create vector data space of INT type.

Parameters

<i>m</i>	Number of rows
<i>u</i>	Pointer to ivector (OUTPUT)

Author

Chensong Zhang

Date

2010/04/06

Definition at line 119 of file vec.c.

9.105.2.12 ivector 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.

9.105.2.13 void fasp_ivec_free (ivec * *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.

9.105.2.14 void fasp_ivec_set (const INT *m*, ivector * *u*)

Set ivector value to be *m*.

Parameters

<i>m</i>	Integer value of ivector
<i>u</i>	Pointer to ivector (MODIFIED)

Author

Chensong Zhang

Date

04/03/2010

Modified by Chunsheng Feng, Xiaoqiang Yue

Date

05/23/2012

Definition at line 304 of file vec.c.

9.106 wrapper.c File Reference

Wrappers for accessing functions by advanced users.

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

Functions

- void `fasp_fwrapper_amg_` (INT *n, INT *nnz, INT *ia, INT *ja, REAL *a, REAL *b, REAL *u, REAL *tol, INT *maxit, INT *ptrlvl)
Solve $Ax=b$ by Ruge and Stuben's classic AMG.
- void `fasp_fwrapper_krylov_amg_` (INT *n, INT *nnz, INT *ia, INT *ja, REAL *a, REAL *b, REAL *u, REAL *tol, INT *maxit, INT *ptrlvl)
Solve $Ax=b$ by Krylov method preconditioned by classic AMG.
- INT `fasp_wrapper_dbsr_krylov_amg` (INT n, INT nnz, INT nb, INT *ia, INT *ja, REAL *a, REAL *b, REAL *u, REAL tol, INT maxit, INT ptrlvl)
Solve $Ax=b$ by Krylov method preconditioned by AMG (dcsr -> dbsr)
- INT `fasp_wrapper_dcoo_dbsr_krylov_amg` (INT n, INT nnz, INT nb, INT *ia, INT *ja, REAL *a, REAL *b, REAL *u, REAL tol, INT maxit, INT ptrlvl)
Solve $Ax=b$ by Krylov method preconditioned by AMG (dcoo -> dbsr)

9.106.1 Detailed Description

Wrappers for accessing functions by advanced users.

Note

Input variables should not need [fasp.h](#)!!!

Definition in file [wrapper.c](#).

9.106.2 Function Documentation

9.106.2.1 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 num 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.

9.106.2.2 void fasp_fwapper_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 num 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.

9.106.2.3 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 num of iterations
<i>ptrlvl</i>	Print level for iterative solvers

Author

Xiaoze Hu

Date

03/05/2013

Definition at line 144 of file wrapper.c.

9.106.2.4 `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 num 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.

Index

__FASP_HEADER__
fasp.h, [145](#)

A

precond_block_data, [49](#)
precond_FASP_blkoi_data, [60](#)

ABS

fasp.h, [145](#)

AMG_ILU_levels

input_param, [39](#)

AMG_aggressive_level

input_param, [39](#)

AMG_aggressive_path

input_param, [39](#)

AMG_amli_degree

input_param, [39](#)

AMG_coarse_dof

input_param, [39](#)

AMG_coarse_scaling

input_param, [39](#)

AMG_coarsening_type

input_param, [39](#)

AMG_cycle_type

input_param, [39](#)

AMG_data, [19](#)

AMG_data_bsr, [20](#)

AMG_interpolation_type

input_param, [40](#)

AMG_levels

input_param, [40](#)

AMG_max_aggregation

input_param, [40](#)

AMG_max_row_sum

input_param, [40](#)

AMG_maxit

input_param, [40](#)

AMG_nl_amli_krylov_type

input_param, [40](#)

AMG_param, [21](#)

AMG_polynomial_degree

input_param, [40](#)

AMG_postsmooth_iter

input_param, [40](#)

AMG_presmooth_iter

input_param, [40](#)

AMG_relaxation

input_param, [41](#)

AMG_schwarz_levels

input_param, [41](#)

AMG_smooth_filter

input_param, [41](#)

AMG_smooth_order

input_param, [41](#)

AMG_smoother

input_param, [41](#)

AMG_strong_coupled

input_param, [41](#)

AMG_strong_threshold

input_param, [41](#)

AMG_tentative_smooth

input_param, [41](#)

AMG_tol

input_param, [41](#)

AMG_truncation_threshold

input_param, [42](#)

AMG_type

input_param, [42](#)

AMLI_CYCLE

messages.h, [234](#)

ASCEND

messages.h, [234](#)

Aarray

precond_block_data, [49](#)

Ablock

precond_block_data, [49](#)

amg.c, [65](#)

fasp_solver_amg, [65](#)

amg_setup_aggregation_bsr.inl, [66](#)

amg_setup_aggregation_csr.inl, [66](#)

amg_setup_cr.c, [66](#)

fasp_amg_setup_cr, [67](#)

amg_setup_rs.c, [67](#)

fasp_amg_setup_rs, [68](#)

fasp_amg_setup_rs_omp, [68](#)

amg_setup_sa.c, [69](#)

fasp_amg_setup_sa, [69](#)

amg_setup_ua.c, [70](#)

fasp_amg_setup_ua, [71](#)

fasp_amg_setup_ua_bsr, [71](#)

amg_solve.c, [71](#)

fasp_amg_solve, [72](#)

- fasp_amg_solve_amli, 72
 - fasp_amg_solve_nl_amli, 73
 - fasp_famg_solve, 73
- amgparam
 - precond_block_data, 49
- amlirecur.c, 74
 - fasp_amg_amli_coef, 75
 - fasp_solver_amli, 76
 - fasp_solver_nl_amli, 76
 - fasp_solver_nl_amli_bsr, 77
- array.c, 77
 - fasp_array_cp, 78
 - fasp_array_cp_nc3, 78
 - fasp_array_cp_nc5, 79
 - fasp_array_cp_nc7, 79
 - fasp_array_null, 79
 - fasp_array_set, 80
 - fasp_iarray_cp, 80
 - fasp_iarray_set, 81
- BIGREAL
 - fasp.h, 145
- blas_array.c, 81
 - fasp_blas_array_ax, 82
 - fasp_blas_array_axpby, 82
 - fasp_blas_array_axpy, 83
 - fasp_blas_array_axpyz, 83
 - fasp_blas_array_dotprod, 84
 - fasp_blas_array_norm1, 84
 - fasp_blas_array_norm2, 85
 - fasp_blas_array_norminf, 85
- blas_bcsr.c, 86
 - fasp_blas_bdbsr_aAxy, 87
 - fasp_blas_bdbsr_mxxv, 88
 - fasp_blas_bdcscr_aAxy, 88
 - fasp_blas_bdcscr_mxxv, 88
- blas_bsr.c, 89
 - fasp_blas_dbsr_aAxpby, 90
 - fasp_blas_dbsr_aAxy, 90
 - fasp_blas_dbsr_aAxy_agg, 91
 - fasp_blas_dbsr_mxxv, 91
 - fasp_blas_dbsr_mxxv_agg, 92
 - fasp_blas_dbsr_rap, 92
 - fasp_blas_dbsr_rap1, 93
 - fasp_blas_dbsr_rap_agg, 93
- blas_csr.c, 94
 - fasp_blas_dcsr_aAxy, 95
 - fasp_blas_dcsr_aAxy_agg, 95
 - fasp_blas_dcsr_add, 97
 - fasp_blas_dcsr_axm, 97
 - fasp_blas_dcsr_mxxm, 98
 - fasp_blas_dcsr_mxxv, 98
 - fasp_blas_dcsr_mxxv_agg, 99
 - fasp_blas_dcsr_ptap, 99
- fasp_blas_dcsr_rap, 100
 - fasp_blas_dcsr_rap4, 100
 - fasp_blas_dcsr_rap_agg, 101
 - fasp_blas_dcsr_rap_agg1, 101
 - fasp_blas_dcsr_vmv, 102
- blas_csrl.c, 102
 - fasp_blas_csrl_mxxv, 103
- blas_smat.c, 103
 - fasp_blas_array_axpy_nc2, 105
 - fasp_blas_array_axpy_nc3, 105
 - fasp_blas_array_axpy_nc5, 106
 - fasp_blas_array_axpy_nc7, 106
 - fasp_blas_array_axpyz_nc2, 106
 - fasp_blas_array_axpyz_nc3, 107
 - fasp_blas_array_axpyz_nc5, 107
 - fasp_blas_array_axpyz_nc7, 108
 - fasp_blas_smat_aAxpby, 108
 - fasp_blas_smat_mul, 109
 - fasp_blas_smat_mul_nc2, 109
 - fasp_blas_smat_mul_nc3, 110
 - fasp_blas_smat_mul_nc5, 110
 - fasp_blas_smat_mul_nc7, 110
 - fasp_blas_smat_mxxv, 111
 - fasp_blas_smat_mxxv_nc2, 111
 - fasp_blas_smat_mxxv_nc3, 111
 - fasp_blas_smat_mxxv_nc5, 113
 - fasp_blas_smat_mxxv_nc7, 113
 - fasp_blas_smat_ymAx, 113
 - fasp_blas_smat_ymAx_nc2, 115
 - fasp_blas_smat_ymAx_nc3, 115
 - fasp_blas_smat_ymAx_nc5, 116
 - fasp_blas_smat_ymAx_nc7, 116
 - fasp_blas_smat_ymAx_ns, 117
 - fasp_blas_smat_ymAx_ns2, 117
 - fasp_blas_smat_ymAx_ns3, 118
 - fasp_blas_smat_ymAx_ns5, 118
 - fasp_blas_smat_ymAx_ns7, 119
 - fasp_blas_smat_ypAx, 119
 - fasp_blas_smat_ypAx_nc2, 120
 - fasp_blas_smat_ypAx_nc3, 120
 - fasp_blas_smat_ypAx_nc5, 120
 - fasp_blas_smat_ypAx_nc7, 122
- blas_str.c, 122
 - fasp_blas_dstr_aAxy, 123
 - fasp_blas_dstr_mxxv, 123
 - fasp_dstr_diagscale, 123
- blas_vec.c, 124
 - fasp_blas_dvec_axpy, 125
 - fasp_blas_dvec_axpyz, 125
 - fasp_blas_dvec_dotprod, 125
 - fasp_blas_dvec_norm1, 126
 - fasp_blas_dvec_norm2, 126
 - fasp_blas_dvec_norminf, 127
 - fasp_blas_dvec_relerr, 127

- block_BSR, 23
 - fasp_block.h, 152
- block_Reservoir, 26
 - fasp_block.h, 153
- block_dCSRmat, 24
 - fasp_block.h, 152
- block_dvector, 25
 - fasp_block.h, 153
- block_iCSRmat, 25
 - fasp_block.h, 153
- block_ivector, 26
 - fasp_block.h, 153
- C2N
 - fasp.h, 145
- CF_ORDER
 - messages.h, 234
- CGPT
 - messages.h, 234
- CLASSIC_AMG
 - messages.h, 234
- COARSE_AC
 - messages.h, 234
- COARSE_CR
 - messages.h, 234
- COARSE_RS
 - messages.h, 235
- CPFIRST
 - messages.h, 235
- checkmat.c, 128
 - fasp_check_dCSRmat, 129
 - fasp_check_diagdom, 129
 - fasp_check_diagpos, 129
 - fasp_check_diagzero, 131
 - fasp_check_iCSRmat, 131
 - fasp_check_symm, 131
- coarsening_cr.c, 133
 - fasp_amg_coarsening_cr, 133
- coarsening_rs.c, 134
 - fasp_amg_coarsening_rs, 135
- col_idx
 - precond_block_data, 49
- convert.c, 136
 - endian_convert_int, 137
 - endian_convert_real, 137
 - fasp_aux_bbyteToldouble, 137
 - fasp_aux_change_endian4, 139
 - fasp_aux_change_endian8, 139
- dBSRmat, 27
 - fasp_block.h, 153
 - JA, 27
 - val, 27
- dCOOmat, 28
 - fasp.h, 149
- dCSRmat, 29
 - fasp.h, 149
- dCSRmat2SAMGInput
 - interface_samg.c, 178
- DESCEND
 - messages.h, 235
- DIAGONAL_PREF
 - fasp.h, 146
- DLMALLOC
 - fasp.h, 146
- dSTRmat, 31
 - fasp.h, 149
- ddenmat, 30
 - fasp.h, 149
- diag
 - precond_block_reservoir_data, 51
- diaginv
 - precond_block_reservoir_data, 51
 - precond_FASP_blkoi_data, 60
- diaginv_S
 - precond_FASP_blkoi_data, 60
- diaginv_noscale
 - precond_FASP_blkoi_data, 60
- diaginvS
 - precond_block_reservoir_data, 51
- doxygen.h, 140
- dvector, 31
 - fasp.h, 149
- dvector2SAMGInput
 - interface_samg.c, 178
- e
 - grid2d, 33
- ERROR_ALLOC_MEM
 - messages.h, 235
- ERROR_AMG_COARSEING
 - messages.h, 235
- ERROR_DATA_ZERODIAG
 - messages.h, 236
- ERROR_DUMMY_VAR
 - messages.h, 236
- ERROR_INPUT_PAR
 - messages.h, 236
- ERROR_LIC_TYPE
 - messages.h, 236
- ERROR_MISC
 - messages.h, 236
- ERROR_NUM_BLOCKS
 - messages.h, 236
- ERROR_OPEN_FILE
 - messages.h, 236
- ERROR_QUAD_DIM

- messages.h, [237](#)
- ERROR_QUAD_TYPE
 - messages.h, [237](#)
- ERROR_REGRESS
 - messages.h, [237](#)
- ERROR_SOLVER_EXIT
 - messages.h, [237](#)
- ERROR_SOLVER_MAXIT
 - messages.h, [237](#)
- ERROR_SOLVER_MISC
 - messages.h, [237](#)
- ERROR_SOLVER_STAG
 - messages.h, [238](#)
- ERROR_SOLVER_TYPE
 - messages.h, [238](#)
- ERROR_WRONG_FILE
 - messages.h, [238](#)
- edges
 - grid2d, [33](#)
- ediri
 - grid2d, [33](#)
- efather
 - grid2d, [33](#)
- eigen.c, [140](#)
 - fasp_dcsr_eig, [140](#)
- endian_convert_int
 - convert.c, [137](#)
- endian_convert_real
 - convert.c, [137](#)
- FALSE
 - messages.h, [238](#)
- FASP_GET_START_END
 - threads.c, [409](#)
- FASP_GSRB
 - fasp.h, [146](#)
- FASP_USE_ILU
 - fasp.h, [146](#)
- FGPT
 - messages.h, [238](#)
- FPPFIRST
 - messages.h, [238](#)
- factor.f, [141](#)
- famg.c, [141](#)
 - fasp_solver_famg, [142](#)
- fasp.h, [142](#)
 - __FASP_HEADER__, [145](#)
 - ABS, [145](#)
 - BIGREAL, [145](#)
 - C2N, [145](#)
 - dCOOmat, [149](#)
 - dCSRmat, [149](#)
 - dCSRmat, [149](#)
 - DIAGONAL_PREF, [146](#)
 - DLMALLOC, [146](#)
 - dSTRmat, [149](#)
 - ddenmat, [149](#)
 - dvector, [149](#)
 - FASP_GSRB, [146](#)
 - FASP_USE_ILU, [146](#)
 - GE, [146](#)
 - GT, [146](#)
 - grid2d, [149](#)
 - iCOOmat, [150](#)
 - iCSRmat, [150](#)
 - IMAP, [150](#)
 - INT, [146](#)
 - ISNAN, [146](#)
 - ISTART, [147](#)
 - idenmat, [150](#)
 - ivector, [150](#)
 - LE, [147](#)
 - LONG, [147](#)
 - LONGLONG, [147](#)
 - LS, [147](#)
 - LinkList, [150](#)
 - ListElement, [150](#)
 - MAX, [147](#)
 - MAX_AMG_LVL, [147](#)
 - MAX_REFINE_LVL, [147](#)
 - MAX_RESTART, [148](#)
 - MAX_STAG, [148](#)
 - MAXIMAP, [150](#)
 - MIN, [148](#)
 - N2C, [148](#)
 - NEDMALLOC, [148](#)
 - nx_rb, [151](#)
 - ny_rb, [151](#)
 - nz_rb, [151](#)
 - OPENMP_HOLDS, [148](#)
 - pcgrid2d, [150](#)
 - pgrid2d, [150](#)
 - REAL, [148](#)
 - RS_C1, [148](#)
 - SHORT, [149](#)
 - SMALLREAL, [149](#)
 - STAG_RATIO, [149](#)
 - total_alloc_count, [151](#)
 - total_alloc_mem, [151](#)
 - fasp_BinarySearch
 - ordering.c, [255](#)
 - fasp_amg_amli_coef
 - amlirecur.c, [75](#)
 - fasp_amg_coarsening_cr
 - coarsening_cr.c, [133](#)
 - fasp_amg_coarsening_rs
 - coarsening_rs.c, [135](#)
 - fasp_amg_data_bsr_create

init.c, [172](#)
 fasp_amg_data_bsr_free
 init.c, [172](#)
 fasp_amg_data_create
 init.c, [172](#)
 fasp_amg_data_free
 init.c, [173](#)
 fasp_amg_interp
 interpolation.c, [182](#)
 fasp_amg_interp1
 interpolation.c, [183](#)
 fasp_amg_interp_em
 interpolation_em.c, [184](#)
 fasp_amg_interp_trunc
 interpolation.c, [183](#)
 fasp_amg_setup_cr
 amg_setup_cr.c, [67](#)
 fasp_amg_setup_rs
 amg_setup_rs.c, [68](#)
 fasp_amg_setup_rs_omp
 amg_setup_rs.c, [68](#)
 fasp_amg_setup_sa
 amg_setup_sa.c, [69](#)
 fasp_amg_setup_ua
 amg_setup_ua.c, [71](#)
 fasp_amg_setup_ua_bsr
 amg_setup_ua.c, [71](#)
 fasp_amg_solve
 amg_solve.c, [72](#)
 fasp_amg_solve_amli
 amg_solve.c, [72](#)
 fasp_amg_solve_nl_amli
 amg_solve.c, [73](#)
 fasp_array_cp
 array.c, [78](#)
 fasp_array_cp_nc3
 array.c, [78](#)
 fasp_array_cp_nc5
 array.c, [79](#)
 fasp_array_cp_nc7
 array.c, [79](#)
 fasp_array_null
 array.c, [79](#)
 fasp_array_set
 array.c, [80](#)
 fasp_aux_bbyteToldouble
 convert.c, [137](#)
 fasp_aux_change_endian4
 convert.c, [139](#)
 fasp_aux_change_endian8
 convert.c, [139](#)
 fasp_aux_dQuickSort
 ordering.c, [251](#)
 fasp_aux_dQuickSortIndex
 ordering.c, [251](#)
 fasp_aux_givens
 givens.c, [159](#)
 fasp_aux_iQuickSort
 ordering.c, [252](#)
 fasp_aux_iQuickSortIndex
 ordering.c, [252](#)
 fasp_aux_merge
 ordering.c, [253](#)
 fasp_aux_msort
 ordering.c, [253](#)
 fasp_aux_unique
 ordering.c, [253](#)
 fasp_blas_array_ax
 blas_array.c, [82](#)
 fasp_blas_array_axpby
 blas_array.c, [82](#)
 fasp_blas_array_axpy
 blas_array.c, [83](#)
 fasp_blas_array_axpy_nc2
 blas_smat.c, [105](#)
 fasp_blas_array_axpy_nc3
 blas_smat.c, [105](#)
 fasp_blas_array_axpy_nc5
 blas_smat.c, [106](#)
 fasp_blas_array_axpy_nc7
 blas_smat.c, [106](#)
 fasp_blas_array_axpyz
 blas_array.c, [83](#)
 fasp_blas_array_axpyz_nc2
 blas_smat.c, [106](#)
 fasp_blas_array_axpyz_nc3
 blas_smat.c, [107](#)
 fasp_blas_array_axpyz_nc5
 blas_smat.c, [107](#)
 fasp_blas_array_axpyz_nc7
 blas_smat.c, [108](#)
 fasp_blas_array_dotprod
 blas_array.c, [84](#)
 fasp_blas_array_norm1
 blas_array.c, [84](#)
 fasp_blas_array_norm2
 blas_array.c, [85](#)
 fasp_blas_array_norminf
 blas_array.c, [85](#)
 fasp_blas_bdsr_aApy
 blas_bcsr.c, [87](#)
 fasp_blas_bdsr_mxv
 blas_bcsr.c, [88](#)
 fasp_blas_bdcsl_aApy
 blas_bcsr.c, [88](#)
 fasp_blas_bdcsl_mxv
 blas_bcsr.c, [88](#)
 fasp_blas_bdsr_aApyby

blas_bsr.c, [90](#)
fasp_blas_dbsr_aAxy
 blas_bsr.c, [90](#)
fasp_blas_dbsr_aAxy_agg
 blas_bsr.c, [91](#)
fasp_blas_dbsr_mxv
 blas_bsr.c, [91](#)
fasp_blas_dbsr_mxv_agg
 blas_bsr.c, [92](#)
fasp_blas_dbsr_rap
 blas_bsr.c, [92](#)
fasp_blas_dbsr_rap1
 blas_bsr.c, [93](#)
fasp_blas_dbsr_rap_agg
 blas_bsr.c, [93](#)
fasp_blas_dcsr_aAxy
 blas_csr.c, [95](#)
fasp_blas_dcsr_aAxy_agg
 blas_csr.c, [95](#)
fasp_blas_dcsr_add
 blas_csr.c, [97](#)
fasp_blas_dcsr_axm
 blas_csr.c, [97](#)
fasp_blas_dcsr_mxm
 blas_csr.c, [98](#)
fasp_blas_dcsr_mxv
 blas_csr.c, [98](#)
fasp_blas_dcsr_mxv_agg
 blas_csr.c, [99](#)
fasp_blas_dcsr_ptap
 blas_csr.c, [99](#)
fasp_blas_dcsr_rap
 blas_csr.c, [100](#)
fasp_blas_dcsr_rap2
 rap.c, [319](#)
fasp_blas_dcsr_rap4
 blas_csr.c, [100](#)
fasp_blas_dcsr_rap_agg
 blas_csr.c, [101](#)
fasp_blas_dcsr_rap_agg1
 blas_csr.c, [101](#)
fasp_blas_dcsr_vmv
 blas_csr.c, [102](#)
fasp_blas_dcsr_mxv
 blas_csr.c, [103](#)
fasp_blas_dstr_aAxy
 blas_str.c, [123](#)
fasp_blas_dstr_mxv
 blas_str.c, [123](#)
fasp_blas_dvec_axpy
 blas_vec.c, [125](#)
fasp_blas_dvec_axpyz
 blas_vec.c, [125](#)
fasp_blas_dvec_dotprod
 blas_vec.c, [125](#)
fasp_blas_dvec_norm1
 blas_vec.c, [126](#)
fasp_blas_dvec_norm2
 blas_vec.c, [126](#)
fasp_blas_dvec_norminf
 blas_vec.c, [127](#)
fasp_blas_dvec_reterr
 blas_vec.c, [127](#)
fasp_blas_smat_aAxyby
 blas_smat.c, [108](#)
fasp_blas_smat_inv
 smat.c, [323](#)
fasp_blas_smat_inv_nc2
 smat.c, [323](#)
fasp_blas_smat_inv_nc3
 smat.c, [323](#)
fasp_blas_smat_inv_nc4
 smat.c, [324](#)
fasp_blas_smat_inv_nc5
 smat.c, [324](#)
fasp_blas_smat_inv_nc7
 smat.c, [324](#)
fasp_blas_smat_mul
 blas_smat.c, [109](#)
fasp_blas_smat_mul_nc2
 blas_smat.c, [109](#)
fasp_blas_smat_mul_nc3
 blas_smat.c, [110](#)
fasp_blas_smat_mul_nc5
 blas_smat.c, [110](#)
fasp_blas_smat_mul_nc7
 blas_smat.c, [110](#)
fasp_blas_smat_mxv
 blas_smat.c, [111](#)
fasp_blas_smat_mxv_nc2
 blas_smat.c, [111](#)
fasp_blas_smat_mxv_nc3
 blas_smat.c, [111](#)
fasp_blas_smat_mxv_nc5
 blas_smat.c, [113](#)
fasp_blas_smat_mxv_nc7
 blas_smat.c, [113](#)
fasp_blas_smat_ymAx
 blas_smat.c, [113](#)
fasp_blas_smat_ymAx_nc2
 blas_smat.c, [115](#)
fasp_blas_smat_ymAx_nc3
 blas_smat.c, [115](#)
fasp_blas_smat_ymAx_nc5
 blas_smat.c, [116](#)
fasp_blas_smat_ymAx_nc7
 blas_smat.c, [116](#)
fasp_blas_smat_ymAx_ns

- blas_smat.c, [117](#)
- fasp_blas_smat_ymAx_ns2
 - blas_smat.c, [117](#)
- fasp_blas_smat_ymAx_ns3
 - blas_smat.c, [118](#)
- fasp_blas_smat_ymAx_ns5
 - blas_smat.c, [118](#)
- fasp_blas_smat_ymAx_ns7
 - blas_smat.c, [119](#)
- fasp_blas_smat_ypAx
 - blas_smat.c, [119](#)
- fasp_blas_smat_ypAx_nc2
 - blas_smat.c, [120](#)
- fasp_blas_smat_ypAx_nc3
 - blas_smat.c, [120](#)
- fasp_blas_smat_ypAx_nc5
 - blas_smat.c, [120](#)
- fasp_blas_smat_ypAx_nc7
 - blas_smat.c, [122](#)
- fasp_block.h, [151](#)
 - block_BSR, [152](#)
 - block_Reservoir, [153](#)
 - block_dCSRmat, [152](#)
 - block_dvector, [153](#)
 - block_iCSRmat, [153](#)
 - block_ivector, [153](#)
 - dBSRmat, [153](#)
 - precond_block_reservoir_data, [153](#)
- fasp_check_dCSRmat
 - checkmat.c, [129](#)
- fasp_check_diagdom
 - checkmat.c, [129](#)
- fasp_check_diagpos
 - checkmat.c, [129](#)
- fasp_check_diagzero
 - checkmat.c, [131](#)
- fasp_check_iCSRmat
 - checkmat.c, [131](#)
- fasp_check_symm
 - checkmat.c, [131](#)
- fasp_chkerr
 - message.c, [228](#)
- fasp_dbsr_alloc
 - sparse_bsr.c, [357](#)
- fasp_dbsr_cp
 - sparse_bsr.c, [358](#)
- fasp_dbsr_create
 - sparse_bsr.c, [358](#)
- fasp_dbsr_diaginv
 - sparse_bsr.c, [359](#)
- fasp_dbsr_diaginv2
 - sparse_bsr.c, [359](#)
- fasp_dbsr_diaginv3
 - sparse_bsr.c, [359](#)
- fasp_dbsr_diaginv4
 - sparse_bsr.c, [360](#)
- fasp_dbsr_diagpref
 - sparse_bsr.c, [360](#)
- fasp_dbsr_free
 - sparse_bsr.c, [361](#)
- fasp_dbsr_getblk
 - sparse_block.c, [355](#)
- fasp_dbsr_getblk_dcsr
 - sparse_block.c, [355](#)
- fasp_dbsr_getdiag
 - sparse_bsr.c, [361](#)
- fasp_dbsr_getdiaginv
 - sparse_bsr.c, [362](#)
- fasp_dbsr_null
 - sparse_bsr.c, [362](#)
- fasp_dbsr_plot
 - graphics.c, [166](#)
- fasp_dbsr_print
 - io.c, [187](#)
- fasp_dbsr_read
 - io.c, [187](#)
- fasp_dbsr_trans
 - sparse_bsr.c, [363](#)
- fasp_dbsr_write
 - io.c, [188](#)
- fasp_dbsr_write_coo
 - io.c, [188](#)
- fasp_dcoo1_read
 - io.c, [189](#)
- fasp_dcoo_alloc
 - sparse_coo.c, [364](#)
- fasp_dcoo_create
 - sparse_coo.c, [364](#)
- fasp_dcoo_free
 - sparse_coo.c, [364](#)
- fasp_dcoo_print
 - io.c, [189](#)
- fasp_dcoo_read
 - io.c, [189](#)
- fasp_dcoo_shift
 - sparse_coo.c, [365](#)
- fasp_dcoo_write
 - io.c, [190](#)
- fasp_dcsr_alloc
 - sparse_csr.c, [367](#)
- fasp_dcsr_compress
 - sparse_csr.c, [367](#)
- fasp_dcsr_compress_inplace
 - sparse_csr.c, [367](#)
- fasp_dcsr_cp
 - sparse_csr.c, [368](#)
- fasp_dcsr_create
 - sparse_csr.c, [368](#)

fasp_dcsr_diagpref
 sparse_csr.c, 369

fasp_dcsr_eig
 eigen.c, 140

fasp_dcsr_free
 sparse_csr.c, 369

fasp_dcsr_getblk
 sparse_block.c, 356

fasp_dcsr_getcol
 sparse_csr.c, 370

fasp_dcsr_getdiag
 sparse_csr.c, 370

fasp_dcsr_multicoloring
 sparse_csr.c, 370

fasp_dcsr_null
 sparse_csr.c, 372

fasp_dcsr_perm
 sparse_csr.c, 372

fasp_dcsr_print
 io.c, 190

fasp_dcsr_read
 io.c, 191

fasp_dcsr_regdiag
 sparse_csr.c, 373

fasp_dcsr_shift
 sparse_csr.c, 373

fasp_dcsr_sort
 sparse_csr.c, 373

fasp_dcsr_subplot
 graphics.c, 166

fasp_dcsr_symdiagscale
 sparse_csr.c, 375

fasp_dcsr_sympat
 sparse_csr.c, 375

fasp_dcsr_trans
 sparse_csr.c, 375

fasp_dcsrl_create
 sparse_csrl.c, 379

fasp_dcsrl_free
 sparse_csrl.c, 380

fasp_dcsrvec1_read
 io.c, 191

fasp_dcsrvec1_write
 io.c, 192

fasp_dcsrvec2_read
 io.c, 192

fasp_dcsrvec2_write
 io.c, 193

fasp_dmtx_read
 io.c, 194

fasp_dmtxsym_read
 io.c, 194

fasp_dstr_alloc
 sparse_str.c, 381

fasp_dstr_cp
 sparse_str.c, 381

fasp_dstr_create
 sparse_str.c, 382

fasp_dstr_diagscale
 blas_str.c, 123

fasp_dstr_free
 sparse_str.c, 382

fasp_dstr_null
 sparse_str.c, 382

fasp_dstr_print
 io.c, 195

fasp_dstr_read
 io.c, 195

fasp_dstr_write
 io.c, 196

fasp_dvec_alloc
 vec.c, 412

fasp_dvec_cp
 vec.c, 412

fasp_dvec_create
 vec.c, 412

fasp_dvec_free
 vec.c, 413

fasp_dvec_isnan
 vec.c, 413

fasp_dvec_maxdiff
 vec.c, 413

fasp_dvec_null
 vec.c, 414

fasp_dvec_print
 io.c, 196

fasp_dvec_rand
 vec.c, 414

fasp_dvec_read
 io.c, 196

fasp_dvec_set
 vec.c, 415

fasp_dvec_symdiagscale
 vec.c, 415

fasp_dvec_write
 io.c, 197

fasp_dvecind_read
 io.c, 197

fasp_dvecind_write
 io.c, 198

fasp_famg_solve
 amg_solve.c, 73

fasp_format_bdcsr_dcsr
 formats.c, 155

fasp_format_dbsr_dcoo
 formats.c, 155

fasp_format_dbsr_dcsr
 formats.c, 156

`fasp_format_dcoo_dcsr`
 formats.c, 156

`fasp_format_dcsr_dbsr`
 formats.c, 156

`fasp_format_dcsr_dcoo`
 formats.c, 157

`fasp_format_dcsr_dcsr`
 formats.c, 157

`fasp_format_dstr_dbsr`
 formats.c, 158

`fasp_format_dstr_dcsr`
 formats.c, 158

`fasp_fwrapper_amg_`
 wrapper.c, 419

`fasp_fwrapper_krylov_amg_`
 wrapper.c, 419

`fasp_gauss2d`
 quadrature.c, 318

`fasp_generate_diaginv_block`
 smoother_str.c, 347

`fasp_gettime`
 timing.c, 410

`fasp_grid2d_plot`
 graphics.c, 167

`fasp_iarray_cp`
 array.c, 80

`fasp_iarray_set`
 array.c, 81

`fasp_icsr_cp`
 sparse_csr.c, 377

`fasp_icsr_create`
 sparse_csr.c, 377

`fasp_icsr_free`
 sparse_csr.c, 378

`fasp_icsr_null`
 sparse_csr.c, 378

`fasp_icsr_trans`
 sparse_csr.c, 378

`fasp_iden_free`
 smat.c, 325

`fasp_ilu_data_alloc`
 init.c, 173

`fasp_ilu_data_free`
 init.c, 173

`fasp_ilu_data_null`
 init.c, 174

`fasp_ilu_dbsr_setup`
 ilu_setup_bsr.c, 168

`fasp_ilu_dcsr_setup`
 ilu_setup_csr.c, 169

`fasp_ilu_dstr_setup0`
 ilu_setup_str.c, 170

`fasp_ilu_dstr_setup1`
 ilu_setup_str.c, 170

`fasp_ivec_alloc`
 vec.c, 416

`fasp_ivec_create`
 vec.c, 416

`fasp_ivec_free`
 vec.c, 416

`fasp_ivec_print`
 io.c, 198

`fasp_ivec_read`
 io.c, 199

`fasp_ivec_set`
 vec.c, 418

`fasp_ivec_write`
 io.c, 199

`fasp_ivecind_read`
 io.c, 200

`fasp_matrix_read`
 io.c, 200

`fasp_matrix_read_bin`
 io.c, 201

`fasp_matrix_write`
 io.c, 201

`fasp_mem_check`
 memory.c, 225

`fasp_mem_dcsr_check`
 memory.c, 225

`fasp_mem_free`
 memory.c, 226

`fasp_mem_iludata_check`
 memory.c, 226

`fasp_mem_realloc`
 memory.c, 227

`fasp_mem_usage`
 memory.c, 227

`fasp_param_amg_init`
 parameters.c, 257

`fasp_param_amg_print`
 parameters.c, 258

`fasp_param_amg_set`
 parameters.c, 258

`fasp_param_amg_to_prec`
 parameters.c, 258

`fasp_param_amg_to_prec_bsr`
 parameters.c, 259

`fasp_param_check`
 input.c, 176

`fasp_param_ilu_init`
 parameters.c, 259

`fasp_param_ilu_print`
 parameters.c, 259

`fasp_param_ilu_set`
 parameters.c, 260

`fasp_param_init`
 parameters.c, 260

fasp_param_input
input.c, [176](#)

fasp_param_input_init
parameters.c, [260](#)

fasp_param_prec_to_amg
parameters.c, [261](#)

fasp_param_prec_to_amg_bsr
parameters.c, [261](#)

fasp_param_schwarz_init
parameters.c, [261](#)

fasp_param_schwarz_print
parameters.c, [263](#)

fasp_param_schwarz_set
parameters.c, [263](#)

fasp_param_set
parameters.c, [263](#)

fasp_param_solver_init
parameters.c, [264](#)

fasp_param_solver_print
parameters.c, [264](#)

fasp_param_solver_set
parameters.c, [264](#)

fasp_poisson_fgmg_1D
gm_g_poisson.c, [160](#)

fasp_poisson_fgmg_2D
gm_g_poisson.c, [161](#)

fasp_poisson_fgmg_3D
gm_g_poisson.c, [161](#)

fasp_poisson_gmg_1D
gm_g_poisson.c, [162](#)

fasp_poisson_gmg_2D
gm_g_poisson.c, [162](#)

fasp_poisson_gmg_3D
gm_g_poisson.c, [163](#)

fasp_poisson_pcg_gmg_1D
gm_g_poisson.c, [163](#)

fasp_poisson_pcg_gmg_2D
gm_g_poisson.c, [164](#)

fasp_poisson_pcg_gmg_3D
gm_g_poisson.c, [164](#)

fasp_precond_amg
precond_csr.c, [297](#)

fasp_precond_amli
precond_csr.c, [297](#)

fasp_precond_data_null
init.c, [174](#)

fasp_precond_dbsr_amg
precond_bsr.c, [290](#)

fasp_precond_dbsr_diag
precond_bsr.c, [291](#)

fasp_precond_dbsr_diag_nc2
precond_bsr.c, [291](#)

fasp_precond_dbsr_diag_nc3
precond_bsr.c, [292](#)

fasp_precond_dbsr_diag_nc5
precond_bsr.c, [292](#)

fasp_precond_dbsr_diag_nc7
precond_bsr.c, [294](#)

fasp_precond_dbsr_ilu
precond_bsr.c, [294](#)

fasp_precond_dbsr_nl_amli
precond_bsr.c, [296](#)

fasp_precond_diag
precond_csr.c, [298](#)

fasp_precond_dstr_blockgs
precond_str.c, [303](#)

fasp_precond_dstr_diag
precond_str.c, [303](#)

fasp_precond_dstr_ilu0
precond_str.c, [304](#)

fasp_precond_dstr_ilu0_backward
precond_str.c, [304](#)

fasp_precond_dstr_ilu0_forward
precond_str.c, [304](#)

fasp_precond_dstr_ilu1
precond_str.c, [306](#)

fasp_precond_dstr_ilu1_backward
precond_str.c, [306](#)

fasp_precond_dstr_ilu1_forward
precond_str.c, [306](#)

fasp_precond_famg
precond_csr.c, [298](#)

fasp_precond_free
precond_csr.c, [299](#)

fasp_precond_ilu
precond_csr.c, [299](#)

fasp_precond_ilu_backward
precond_csr.c, [299](#)

fasp_precond_ilu_forward
precond_csr.c, [300](#)

fasp_precond_nl_amli
precond_csr.c, [300](#)

fasp_precond_null
init.c, [174](#)

fasp_precond_schwarz
precond_csr.c, [300](#)

fasp_precond_setup
precond_csr.c, [302](#)

fasp_quad2d
quadrature.c, [318](#)

fasp_schwarz_data_free
init.c, [175](#)

fasp_schwarz_setup
schwarz_setup.c, [321](#)

fasp_set_GS_threads
itsolver_bsr.c, [207](#)

fasp_smat_identity
smat.c, [325](#)

fasp_smat_identity_nc2
 smat.c, [325](#)
 fasp_smat_identity_nc3
 smat.c, [327](#)
 fasp_smat_identity_nc5
 smat.c, [327](#)
 fasp_smat_identity_nc7
 smat.c, [327](#)
 fasp_smat_lu_decomp
 lu.c, [223](#)
 fasp_smat_lu_solve
 lu.c, [224](#)
 fasp_smoother_dbsr_gs
 smoother_bsr.c, [329](#)
 fasp_smoother_dbsr_gs1
 smoother_bsr.c, [329](#)
 fasp_smoother_dbsr_gs_ascend
 smoother_bsr.c, [330](#)
 fasp_smoother_dbsr_gs_ascend1
 smoother_bsr.c, [330](#)
 fasp_smoother_dbsr_gs_descend
 smoother_bsr.c, [331](#)
 fasp_smoother_dbsr_gs_descend1
 smoother_bsr.c, [331](#)
 fasp_smoother_dbsr_gs_order1
 smoother_bsr.c, [332](#)
 fasp_smoother_dbsr_gs_order2
 smoother_bsr.c, [332](#)
 fasp_smoother_dbsr_ilu
 smoother_bsr.c, [333](#)
 fasp_smoother_dbsr_jacobi
 smoother_bsr.c, [333](#)
 fasp_smoother_dbsr_jacobi1
 smoother_bsr.c, [333](#)
 fasp_smoother_dbsr_jacobi_setup
 smoother_bsr.c, [335](#)
 fasp_smoother_dbsr_sor
 smoother_bsr.c, [335](#)
 fasp_smoother_dbsr_sor1
 smoother_bsr.c, [336](#)
 fasp_smoother_dbsr_sor_ascend
 smoother_bsr.c, [336](#)
 fasp_smoother_dbsr_sor_descend
 smoother_bsr.c, [337](#)
 fasp_smoother_dbsr_sor_order
 smoother_bsr.c, [337](#)
 fasp_smoother_dcsr_L1diag
 smoother_csr.c, [341](#)
 fasp_smoother_dcsr_gs
 smoother_csr.c, [339](#)
 fasp_smoother_dcsr_gs_cf
 smoother_csr.c, [339](#)
 fasp_smoother_dcsr_gscr
 smoother_csr_cr.c, [344](#)
 fasp_smoother_dcsr_ilu
 smoother_csr.c, [340](#)
 fasp_smoother_dcsr_jacobi
 smoother_csr.c, [340](#)
 fasp_smoother_dcsr_kaczmarz
 smoother_csr.c, [341](#)
 fasp_smoother_dcsr_poly
 smoother_csr_poly.c, [345](#)
 fasp_smoother_dcsr_poly_old
 smoother_csr_poly.c, [345](#)
 fasp_smoother_dcsr_sgs
 smoother_csr.c, [342](#)
 fasp_smoother_dcsr_sor
 smoother_csr.c, [342](#)
 fasp_smoother_dcsr_sor_cf
 smoother_csr.c, [343](#)
 fasp_smoother_dstr_gs
 smoother_str.c, [347](#)
 fasp_smoother_dstr_gs1
 smoother_str.c, [348](#)
 fasp_smoother_dstr_gs_ascend
 smoother_str.c, [348](#)
 fasp_smoother_dstr_gs_cf
 smoother_str.c, [349](#)
 fasp_smoother_dstr_gs_descend
 smoother_str.c, [349](#)
 fasp_smoother_dstr_gs_order
 smoother_str.c, [350](#)
 fasp_smoother_dstr_jacobi
 smoother_str.c, [350](#)
 fasp_smoother_dstr_jacobi1
 smoother_str.c, [351](#)
 fasp_smoother_dstr_schwarz
 smoother_str.c, [351](#)
 fasp_smoother_dstr_sor
 smoother_str.c, [351](#)
 fasp_smoother_dstr_sor1
 smoother_str.c, [352](#)
 fasp_smoother_dstr_sor_ascend
 smoother_str.c, [352](#)
 fasp_smoother_dstr_sor_cf
 smoother_str.c, [353](#)
 fasp_smoother_dstr_sor_descend
 smoother_str.c, [353](#)
 fasp_smoother_dstr_sor_order
 smoother_str.c, [354](#)
 fasp_solver_amg
 amg.c, [65](#)
 fasp_solver_aml
 amlirecur.c, [76](#)
 fasp_solver_bdcscr_itsolver
 itsolver_bcsr.c, [205](#)
 fasp_solver_bdcscr_krylov
 itsolver_bcsr.c, [205](#)

fasp_solver_bdcsr_pbcgs
pbcgs.c, [266](#)

fasp_solver_bdcsr_pcg
pcg.c, [273](#)

fasp_solver_bdcsr_pgmres
pgmres.c, [280](#)

fasp_solver_bdcsr_pminres
pminres.c, [286](#)

fasp_solver_bdcsr_pvgmres
pvgmres.c, [313](#)

fasp_solver_bdcsr_spbcs
spbcgs.c, [392](#)

fasp_solver_bdcsr_spcg
spcg.c, [397](#)

fasp_solver_bdcsr_spgmres
spgmres.c, [400](#)

fasp_solver_bdcsr_spmminres
spmminres.c, [404](#)

fasp_solver_bdcsr_spvgmres
spvgmres.c, [406](#)

fasp_solver_dbsr_itsolver
itsolver_bsr.c, [207](#)

fasp_solver_dbsr_krylov
itsolver_bsr.c, [208](#)

fasp_solver_dbsr_krylov_amg
itsolver_bsr.c, [208](#)

fasp_solver_dbsr_krylov_diag
itsolver_bsr.c, [209](#)

fasp_solver_dbsr_krylov_ilu
itsolver_bsr.c, [209](#)

fasp_solver_dbsr_pbcgs
pbcgs.c, [267](#)

fasp_solver_dbsr_pgmres
pgmres.c, [281](#)

fasp_solver_dbsr_pvgmres
pvgmres.c, [309](#)

fasp_solver_dbsr_pvgmres
pvgmres.c, [313](#)

fasp_solver_dbsr_spbcs
spbcgs.c, [393](#)

fasp_solver_dbsr_spgmres
spgmres.c, [400](#)

fasp_solver_dbsr_spvgmres
spvgmres.c, [407](#)

fasp_solver_dcsr_itsolver
itsolver_csr.c, [211](#)

fasp_solver_dcsr_krylov
itsolver_csr.c, [212](#)

fasp_solver_dcsr_krylov_amg
itsolver_csr.c, [212](#)

fasp_solver_dcsr_krylov_diag
itsolver_csr.c, [213](#)

fasp_solver_dcsr_krylov_ilu
itsolver_csr.c, [213](#)

fasp_solver_dcsr_krylov_ilu_M
itsolver_csr.c, [214](#)

fasp_solver_dcsr_krylov_schwarz
itsolver_csr.c, [214](#)

fasp_solver_dcsr_pbcgs
pbcgs.c, [268](#)

fasp_solver_dcsr_pcg
pcg.c, [274](#)

fasp_solver_dcsr_pgcg
pgcg.c, [278](#)

fasp_solver_dcsr_pgmres
pgmres.c, [282](#)

fasp_solver_dcsr_pminres
pminres.c, [286](#)

fasp_solver_dcsr_pvgmres
pvgmres.c, [310](#)

fasp_solver_dcsr_pvgmres
pvgmres.c, [315](#)

fasp_solver_dcsr_spbcs
spbcgs.c, [393](#)

fasp_solver_dcsr_spcg
spcg.c, [398](#)

fasp_solver_dcsr_spgmres
spgmres.c, [401](#)

fasp_solver_dcsr_spmminres
spmminres.c, [404](#)

fasp_solver_dcsr_spvgmres
spvgmres.c, [407](#)

fasp_solver_dstr_itsolver
itsolver_str.c, [219](#)

fasp_solver_dstr_krylov
itsolver_str.c, [219](#)

fasp_solver_dstr_krylov_blockgs
itsolver_str.c, [220](#)

fasp_solver_dstr_krylov_diag
itsolver_str.c, [220](#)

fasp_solver_dstr_krylov_ilu
itsolver_str.c, [220](#)

fasp_solver_dstr_pbcgs
pbcgs.c, [268](#)

fasp_solver_dstr_pcg
pcg.c, [274](#)

fasp_solver_dstr_pgmres
pgmres.c, [282](#)

fasp_solver_dstr_pminres
pminres.c, [287](#)

fasp_solver_dstr_pvgmres
pvgmres.c, [316](#)

fasp_solver_dstr_spbcs
spbcgs.c, [395](#)

fasp_solver_dstr_spcg
spcg.c, [398](#)

fasp_solver_dstr_spgmres
spgmres.c, [401](#)

- fasp_poisson_pcg_gmg_3D, 164
- gmg_util.inl, 165
- graphics.c, 165
 - fasp_dbsr_plot, 166
 - fasp_dcsr_subplot, 166
 - fasp_grid2d_plot, 167
- grid2d, 32
 - e, 33
 - edges, 33
 - ediri, 33
 - efather, 33
 - fasp.h, 149
 - p, 33
 - pdiri, 33
 - pfather, 33
 - s, 33
 - t, 33
 - tfather, 34
 - triangles, 34
 - vertices, 34
- iCOOmat, 34
 - fasp.h, 150
- iCSRmat, 35
 - fasp.h, 150
- ILU_data, 36
- ILU_droptol
 - input_param, 42
- ILU_lfil
 - input_param, 42
- ILU_param, 37
- ILU_permtol
 - input_param, 42
- ILU_relax
 - input_param, 42
- ILU_type
 - input_param, 42
- ILUk
 - messages.h, 238
- ILUt
 - messages.h, 239
- ILUtp
 - messages.h, 239
- IMAP
 - fasp.h, 150
- INT
 - fasp.h, 146
- INTERP_DIR
 - messages.h, 239
- INTERP_ENG
 - messages.h, 239
- INTERP_STD
 - messages.h, 239
- ISNAN
 - fasp.h, 146
- ISPT
 - messages.h, 239
- ISTART
 - fasp.h, 147
- idenmat, 36
 - fasp.h, 150
- ilu.f, 167
- ilu_setup_bsr.c, 168
 - fasp_ilu_dbsr_setup, 168
- ilu_setup_csr.c, 169
 - fasp_ilu_dcsr_setup, 169
- ilu_setup_str.c, 169
 - fasp_ilu_dstr_setup0, 170
 - fasp_ilu_dstr_setup1, 170
- inifile
 - input_param, 42
- init.c, 171
 - fasp_amg_data_bsr_create, 172
 - fasp_amg_data_bsr_free, 172
 - fasp_amg_data_create, 172
 - fasp_amg_data_free, 173
 - fasp_ilu_data_alloc, 173
 - fasp_ilu_data_free, 173
 - fasp_ilu_data_null, 174
 - fasp_precond_data_null, 174
 - fasp_precond_null, 174
 - fasp_schwarz_data_free, 175
- input.c, 175
 - fasp_param_check, 176
 - fasp_param_input, 176
- input_param, 37
 - AMG_ILU_levels, 39
 - AMG_aggressive_level, 39
 - AMG_aggressive_path, 39
 - AMG_amli_degree, 39
 - AMG_coarse_dof, 39
 - AMG_coarse_scaling, 39
 - AMG_coarsening_type, 39
 - AMG_cycle_type, 39
 - AMG_interpolation_type, 40
 - AMG_levels, 40
 - AMG_max_aggregation, 40
 - AMG_max_row_sum, 40
 - AMG_maxit, 40
 - AMG_nl_amli_krylov_type, 40
 - AMG_polynomial_degree, 40
 - AMG_postsmooth_iter, 40
 - AMG_presmooth_iter, 40
 - AMG_relaxation, 41
 - AMG_schwarz_levels, 41
 - AMG_smooth_filter, 41
 - AMG_smooth_order, 41
 - AMG_smoother, 41

- AMG_strong_coupled, [41](#)
- AMG_strong_threshold, [41](#)
- AMG_tentative_smooth, [41](#)
- AMG_tol, [41](#)
- AMG_truncation_threshold, [42](#)
- AMG_type, [42](#)
- ILU_droptol, [42](#)
- ILU_lfil, [42](#)
- ILU_permtol, [42](#)
- ILU_relax, [42](#)
- ILU_type, [42](#)
- inifile, [42](#)
- itsolver_maxit, [42](#)
- itsolver_tol, [43](#)
- output_type, [43](#)
- precond_type, [43](#)
- print_level, [43](#)
- problem_num, [43](#)
- restart, [43](#)
- Schwarz_maxlvl, [43](#)
- Schwarz_mmsize, [43](#)
- Schwarz_type, [43](#)
- solver_type, [44](#)
- stop_type, [44](#)
- workdir, [44](#)
- interface_mumps.c, [176](#)
 - fasp_solver_mumps, [177](#)
 - fasp_solver_mumps_steps, [177](#)
- interface_samg.c, [178](#)
 - dCSRmat2SAMGInput, [178](#)
 - dvector2SAMGInput, [178](#)
- interface_superlu.c, [180](#)
 - fasp_solver_superlu, [180](#)
- interface_umfpack.c, [181](#)
 - fasp_solver_umfpack, [181](#)
- interpolation.c, [182](#)
 - fasp_amg_interp, [182](#)
 - fasp_amg_interp1, [183](#)
 - fasp_amg_interp_trunc, [183](#)
- interpolation_em.c, [184](#)
 - fasp_amg_interp_em, [184](#)
- io.c, [185](#)
 - fasp_dbsr_print, [187](#)
 - fasp_dbsr_read, [187](#)
 - fasp_dbsr_write, [188](#)
 - fasp_dbsr_write_coo, [188](#)
 - fasp_dcoo1_read, [189](#)
 - fasp_dcoo_print, [189](#)
 - fasp_dcoo_read, [189](#)
 - fasp_dcoo_write, [190](#)
 - fasp_dcsr_print, [190](#)
 - fasp_dcsr_read, [191](#)
 - fasp_dcsrvec1_read, [191](#)
 - fasp_dcsrvec1_write, [192](#)
 - fasp_dcsrvec2_read, [192](#)
 - fasp_dcsrvec2_write, [193](#)
 - fasp_dmtx_read, [194](#)
 - fasp_dmtxsym_read, [194](#)
 - fasp_dstr_print, [195](#)
 - fasp_dstr_read, [195](#)
 - fasp_dstr_write, [196](#)
 - fasp_dvec_print, [196](#)
 - fasp_dvec_read, [196](#)
 - fasp_dvec_write, [197](#)
 - fasp_dvecind_read, [197](#)
 - fasp_dvecind_write, [198](#)
 - fasp_ivec_print, [198](#)
 - fasp_ivec_read, [199](#)
 - fasp_ivec_write, [199](#)
 - fasp_ivecind_read, [200](#)
 - fasp_matrix_read, [200](#)
 - fasp_matrix_read_bin, [201](#)
 - fasp_matrix_write, [201](#)
 - fasp_vector_read, [203](#)
 - fasp_vector_write, [204](#)
- itsolver_bcsr.c, [205](#)
 - fasp_solver_bdcsr_itsolver, [205](#)
 - fasp_solver_bdcsr_krylov, [205](#)
- itsolver_bsr.c, [206](#)
 - fasp_set_GS_threads, [207](#)
 - fasp_solver_dbsr_itsolver, [207](#)
 - fasp_solver_dbsr_krylov, [208](#)
 - fasp_solver_dbsr_krylov_amg, [208](#)
 - fasp_solver_dbsr_krylov_diag, [209](#)
 - fasp_solver_dbsr_krylov_ilu, [209](#)
 - THDs_AMG_GS, [210](#)
 - THDs_CPR_gGS, [210](#)
 - THDs_CPR_IGS, [210](#)
- itsolver_csr.c, [210](#)
 - fasp_solver_dcsr_itsolver, [211](#)
 - fasp_solver_dcsr_krylov, [212](#)
 - fasp_solver_dcsr_krylov_amg, [212](#)
 - fasp_solver_dcsr_krylov_diag, [213](#)
 - fasp_solver_dcsr_krylov_ilu, [213](#)
 - fasp_solver_dcsr_krylov_ilu_M, [214](#)
 - fasp_solver_dcsr_krylov_schwarz, [214](#)
- itsolver_maxit
 - input_param, [42](#)
- itsolver_mf.c, [215](#)
 - fasp_solver_itsolver, [216](#)
 - fasp_solver_itsolver_init, [217](#)
 - fasp_solver_krylov, [217](#)
- itsolver_param, [44](#)
 - itsolver_type, [45](#)
 - maxit, [45](#)
 - precond_type, [45](#)
 - print_level, [45](#)
 - restart, [45](#)

- stop_type, 45
- tol, 45
- itsolver_str.c, 218
 - fasp_solver_dstr_itsolver, 219
 - fasp_solver_dstr_krylov, 219
 - fasp_solver_dstr_krylov_blockgs, 220
 - fasp_solver_dstr_krylov_diag, 220
 - fasp_solver_dstr_krylov_ilu, 220
- itsolver_tol
 - input_param, 43
- itsolver_type
 - itsolver_param, 45
- itsolver_util.inl, 221
- ivector, 45
 - fasp.h, 150
- JA
 - dBSRmat, 27
- LE
 - fasp.h, 147
- LIST_HEAD
 - linklist.inl, 222
- LIST_TAIL
 - linklist.inl, 222
- LONG
 - fasp.h, 147
- LONGLONG
 - fasp.h, 147
- LS
 - fasp.h, 147
- Link, 46
- LinkedList
 - fasp.h, 150
- linked_list, 46
- linklist.inl, 222
 - LIST_HEAD, 222
 - LIST_TAIL, 222
- ListElement
 - fasp.h, 150
- lu.c, 223
 - fasp_smat_lu_decomp, 223
 - fasp_smat_lu_solve, 224
- MAT_BSR
 - messages.h, 239
- MAT_CSR
 - messages.h, 240
- MAT_CSRL
 - messages.h, 240
- MAT_FREE
 - messages.h, 240
- MAT_STR
 - messages.h, 240
- MAT_SymCSR
 - messages.h, 240
- MAT_bBSR
 - messages.h, 239
- MAT_bCSR
 - messages.h, 239
- MAX
 - fasp.h, 147
- MAX_AMG_LVL
 - fasp.h, 147
- MAX_REFINE_LVL
 - fasp.h, 147
- MAX_RESTART
 - fasp.h, 148
- MAX_STAG
 - fasp.h, 148
- MAXIMAP
 - fasp.h, 150
- MIN
 - fasp.h, 148
- maxit
 - itsolver_param, 45
 - precond_FASP_blkoi_data, 60
- memory.c, 224
 - fasp_mem_check, 225
 - fasp_mem_dcsr_check, 225
 - fasp_mem_free, 226
 - fasp_mem_iludata_check, 226
 - fasp_mem_realloc, 227
 - fasp_mem_usage, 227
 - total_alloc_count, 227
 - total_alloc_mem, 227
- message.c, 228
 - fasp_chkerr, 228
 - print_amgcomplexity, 229
 - print_amgcomplexity_bsr, 229
 - print_cputime, 229
 - print_itinfo, 230
 - print_message, 230
- messages.h, 231
 - AMLI_CYCLE, 234
 - ASCEND, 234
 - CF_ORDER, 234
 - CGPT, 234
 - CLASSIC_AMG, 234
 - COARSE_AC, 234
 - COARSE_CR, 234
 - COARSE_RS, 235
 - CPFIRST, 235
 - DESCEND, 235
 - ERROR_ALLOC_MEM, 235
 - ERROR_AMG_COARSEING, 235
 - ERROR_DATA_ZERODIAG, 236
 - ERROR_DUMMY_VAR, 236
 - ERROR_INPUT_PAR, 236

ERROR_LIC_TYPE, [236](#)
 ERROR_MISC, [236](#)
 ERROR_NUM_BLOCKS, [236](#)
 ERROR_OPEN_FILE, [236](#)
 ERROR_QUAD_DIM, [237](#)
 ERROR_QUAD_TYPE, [237](#)
 ERROR_REGRESS, [237](#)
 ERROR_SOLVER_EXIT, [237](#)
 ERROR_SOLVER_MAXIT, [237](#)
 ERROR_SOLVER_MISC, [237](#)
 ERROR_SOLVER_STAG, [238](#)
 ERROR_SOLVER_TYPE, [238](#)
 ERROR_WRONG_FILE, [238](#)
 FALSE, [238](#)
 FGPT, [238](#)
 FPFIRST, [238](#)
 ILUk, [238](#)
 ILU_t, [239](#)
 ILU_{tp}, [239](#)
 INTERP_DIR, [239](#)
 INTERP_ENG, [239](#)
 INTERP_STD, [239](#)
 ISPT, [239](#)
 MAT_BSR, [239](#)
 MAT_CSR, [240](#)
 MAT_CSRL, [240](#)
 MAT_FREE, [240](#)
 MAT_STR, [240](#)
 MAT_SymCSR, [240](#)
 MAT_bBSR, [239](#)
 MAT_bCSR, [239](#)
 NL_AMLI_CYCLE, [240](#)
 NO_ORDER, [240](#)
 OFF, [240](#)
 ON, [241](#)
 PREC_AMG, [241](#)
 PREC_DIAG, [241](#)
 PREC_FMG, [241](#)
 PREC_ILU, [241](#)
 PREC_NULL, [241](#)
 PREC_SCHWARZ, [241](#)
 PRINT_ALL, [241](#)
 PRINT_MIN, [242](#)
 PRINT_MORE, [242](#)
 PRINT_MOST, [242](#)
 PRINT_NONE, [242](#)
 PRINT_SOME, [242](#)
 RUN_FAIL, [242](#)
 SA_AMG, [242](#)
 SMOOTHER_BLKOIL, [242](#)
 SMOOTHER_CG, [243](#)
 SMOOTHER_GS, [243](#)
 SMOOTHER_GSOR, [243](#)
 SMOOTHER_JACOBI, [243](#)
 SMOOTHER_L1DIAG, [243](#)
 SMOOTHER_POLY, [243](#)
 SMOOTHER_SGS, [243](#)
 SMOOTHER_SGSOR, [243](#)
 SMOOTHER_SOR, [244](#)
 SMOOTHER_SPETEN, [244](#)
 SMOOTHER_SSOR, [244](#)
 SOLVER_AMG, [244](#)
 SOLVER_BiCGstab, [244](#)
 SOLVER_CG, [244](#)
 SOLVER_FMG, [244](#)
 SOLVER_GCG, [244](#)
 SOLVER_GMRES, [245](#)
 SOLVER_MUMPS, [245](#)
 SOLVER_MinRes, [245](#)
 SOLVER_SBiCGstab, [245](#)
 SOLVER_SCG, [245](#)
 SOLVER_SGCG, [245](#)
 SOLVER_SGMRES, [245](#)
 SOLVER_SMinRes, [245](#)
 SOLVER_SUPERLU, [245](#)
 SOLVER_SVFGMRES, [246](#)
 SOLVER_SVGMRES, [246](#)
 SOLVER_UMFPACK, [246](#)
 SOLVER_VFGMRES, [246](#)
 SOLVER_VGMRES, [246](#)
 STOP_MOD_REL_RES, [246](#)
 STOP_REL_PRECRES, [246](#)
 STOP_REL_RES, [246](#)
 SUCCESS, [247](#)
 TRUE, [247](#)
 UA_AMG, [247](#)
 UNPT, [247](#)
 USERDEFINED, [247](#)
 V_CYCLE, [247](#)
 W_CYCLE, [247](#)
 mg_util.inl, [248](#)
 mgcycle.c, [248](#)
 fasp_solver_mgcycle, [248](#)
 fasp_solver_mgcycle_bsr, [249](#)
 mgl_data
 precond_FASP_blkoil_data, [60](#)
 mgrecur.c, [249](#)
 fasp_solver_mgrecur, [250](#)
 mxv_matfree, [47](#)

 N2C
 fasp.h, [148](#)
 NEDMALLOC
 fasp.h, [148](#)
 NL_AMLI_CYCLE
 messages.h, [240](#)
 NO_ORDER
 messages.h, [240](#)

- neigh
 - precond_FASP_blkoi_data, 60
- nx_rb
 - fasp.h, 151
- ny_rb
 - fasp.h, 151
- nz_rb
 - fasp.h, 151
- OFF
 - messages.h, 240
- ON
 - messages.h, 241
- OPENMP_HOLDS
 - fasp.h, 148
- order
 - precond_block_reservoir_data, 51
 - precond_FASP_blkoi_data, 61
- ordering.c, 250
 - fasp_BinarySearch, 255
 - fasp_aux_dQuickSort, 251
 - fasp_aux_dQuickSortIndex, 251
 - fasp_aux_iQuickSort, 252
 - fasp_aux_iQuickSortIndex, 252
 - fasp_aux_merge, 253
 - fasp_aux_msort, 253
 - fasp_aux_unique, 253
- output_type
 - input_param, 43
- p
 - grid2d, 33
- PP
 - precond_block_reservoir_data, 52
 - precond_FASP_blkoi_data, 61
- PREC_AMG
 - messages.h, 241
- PREC_DIAG
 - messages.h, 241
- PREC_FMG
 - messages.h, 241
- PREC_ILU
 - messages.h, 241
- PREC_NULL
 - messages.h, 241
- PREC_SCHWARZ
 - messages.h, 241
- PRINT_ALL
 - messages.h, 241
- PRINT_MIN
 - messages.h, 242
- PRINT_MORE
 - messages.h, 242
- PRINT_MOST
 - messages.h, 242
- PRINT_NONE
 - messages.h, 242
- PRINT_SOME
 - messages.h, 242
- parameters.c, 255
 - fasp_param_amg_init, 257
 - fasp_param_amg_print, 258
 - fasp_param_amg_set, 258
 - fasp_param_amg_to_prec, 258
 - fasp_param_amg_to_prec_bsr, 259
 - fasp_param_ilu_init, 259
 - fasp_param_ilu_print, 259
 - fasp_param_ilu_set, 260
 - fasp_param_init, 260
 - fasp_param_input_init, 260
 - fasp_param_prec_to_amg, 261
 - fasp_param_prec_to_amg_bsr, 261
 - fasp_param_schwarz_init, 261
 - fasp_param_schwarz_print, 263
 - fasp_param_schwarz_set, 263
 - fasp_param_set, 263
 - fasp_param_solver_init, 264
 - fasp_param_solver_print, 264
 - fasp_param_solver_set, 264
- pbcgs.c, 265
 - fasp_solver_bdcsr_pbcgs, 266
 - fasp_solver_dbsr_pbcgs, 267
 - fasp_solver_dcsr_pbcgs, 268
 - fasp_solver_dstr_pbcgs, 268
- pbcgs_mf.c, 269
 - fasp_solver_pbcgs, 270
- pcg.c, 271
 - fasp_solver_bdcsr_pcg, 273
 - fasp_solver_dcsr_pcg, 274
 - fasp_solver_dstr_pcg, 274
- pcg_mf.c, 275
 - fasp_solver_pcg, 276
- pcgrid2d
 - fasp.h, 150
- pdiri
 - grid2d, 33
- perf_idx
 - precond_block_reservoir_data, 51
 - precond_FASP_blkoi_data, 61
- perf_neigh
 - precond_FASP_blkoi_data, 61
- pfather
 - grid2d, 33
- pgcg.c, 277
 - fasp_solver_dcsr_pgcg, 278
- pgcg_mf.c, 278
 - fasp_solver_pgcg, 279
- pgmres.c, 280
 - fasp_solver_bdcsr_pgmres, 280

- message.c, [229](#)
- print_cputime
 - message.c, [229](#)
- print_itinfo
 - message.c, [230](#)
- print_level
 - input_param, [43](#)
 - itsolver_param, [45](#)
- print_message
 - message.c, [230](#)
- problem_num
 - input_param, [43](#)
- pvfgmres.c, [308](#)
 - fasp_solver_dbsr_pvfgmres, [309](#)
 - fasp_solver_dcsr_pvfgmres, [310](#)
- pvfgmres_mf.c, [311](#)
 - fasp_solver_pvfgmres, [311](#)
- pvgmres.c, [312](#)
 - fasp_solver_bdcsr_pvgmres, [313](#)
 - fasp_solver_dbsr_pvgmres, [313](#)
 - fasp_solver_dcsr_pvgmres, [315](#)
 - fasp_solver_dstr_pvgmres, [316](#)
- pvgmres_mf.c, [316](#)
 - fasp_solver_pvgmres, [317](#)
- quadrature.c, [318](#)
 - fasp_gauss2d, [318](#)
 - fasp_quad2d, [318](#)
- r
 - precond_block_data, [49](#)
 - precond_block_reservoir_data, [52](#)
 - precond_FASP_blkoi_data, [61](#)
- REAL
 - fasp.h, [148](#)
- RR
 - precond_block_reservoir_data, [52](#)
 - precond_FASP_blkoi_data, [61](#)
- RS_C1
 - fasp.h, [148](#)
- RUN_FAIL
 - messages.h, [242](#)
- rap.c, [319](#)
 - fasp_blas_dcsr_rap2, [319](#)
- restart
 - input_param, [43](#)
 - itsolver_param, [45](#)
 - precond_FASP_blkoi_data, [61](#)
- row_idx
 - precond_block_data, [49](#)
- s
 - grid2d, [33](#)
- SA_AMG
 - messages.h, [242](#)
- SHORT
 - fasp.h, [149](#)
- SMALLREAL
 - fasp.h, [149](#)
- SMOOTHER_BLKOil
 - messages.h, [242](#)
- SMOOTHER_CG
 - messages.h, [243](#)
- SMOOTHER_GS
 - messages.h, [243](#)
- SMOOTHER_GSOR
 - messages.h, [243](#)
- SMOOTHER_JACOBI
 - messages.h, [243](#)
- SMOOTHER_L1DIAG
 - messages.h, [243](#)
- SMOOTHER_POLY
 - messages.h, [243](#)
- SMOOTHER_SGS
 - messages.h, [243](#)
- SMOOTHER_SGSOR
 - messages.h, [243](#)
- SMOOTHER_SOR
 - messages.h, [244](#)
- SMOOTHER_SPETEN
 - messages.h, [244](#)
- SMOOTHER_SSOR
 - messages.h, [244](#)
- SOLVER_AMG
 - messages.h, [244](#)
- SOLVER_BiCGstab
 - messages.h, [244](#)
- SOLVER_CG
 - messages.h, [244](#)
- SOLVER_FMG
 - messages.h, [244](#)
- SOLVER_GCG
 - messages.h, [244](#)
- SOLVER_GMRES
 - messages.h, [245](#)
- SOLVER_MUMPS
 - messages.h, [245](#)
- SOLVER_MinRes
 - messages.h, [245](#)
- SOLVER_SBiCGstab
 - messages.h, [245](#)
- SOLVER_SCG
 - messages.h, [245](#)
- SOLVER_SGCG
 - messages.h, [245](#)
- SOLVER_SGMRES
 - messages.h, [245](#)
- SOLVER_SMinRes
 - messages.h, [245](#)

- SOLVER_SUPERLU
 - messages.h, [245](#)
- SOLVER_SVFGMRES
 - messages.h, [246](#)
- SOLVER_SVGMRES
 - messages.h, [246](#)
- SOLVER_UMFPACK
 - messages.h, [246](#)
- SOLVER_VFGMRES
 - messages.h, [246](#)
- SOLVER_VGMRES
 - messages.h, [246](#)
- SS
 - precond_block_reservoir_data, [52](#)
 - precond_FASP_blkoi_data, [62](#)
- STAG_RATIO
 - fasp.h, [149](#)
- STOP_MOD_REL_RES
 - messages.h, [246](#)
- STOP_REL_PRECRES
 - messages.h, [246](#)
- STOP_REL_RES
 - messages.h, [246](#)
- SUCCESS
 - messages.h, [247](#)
- scaled
 - precond_block_reservoir_data, [52](#)
 - precond_FASP_blkoi_data, [62](#)
- schwarz.f, [320](#)
- Schwarz_data, [62](#)
- Schwarz_maxlvl
 - input_param, [43](#)
- Schwarz_mmsize
 - input_param, [43](#)
- Schwarz_param, [63](#)
- schwarz_setup.c, [321](#)
 - fasp_schwarz_setup, [321](#)
- Schwarz_type
 - input_param, [43](#)
- smat.c, [322](#)
 - fasp_blas_smat_inv, [323](#)
 - fasp_blas_smat_inv_nc2, [323](#)
 - fasp_blas_smat_inv_nc3, [323](#)
 - fasp_blas_smat_inv_nc4, [324](#)
 - fasp_blas_smat_inv_nc5, [324](#)
 - fasp_blas_smat_inv_nc7, [324](#)
 - fasp_iden_free, [325](#)
 - fasp_smat_identity, [325](#)
 - fasp_smat_identity_nc2, [325](#)
 - fasp_smat_identity_nc3, [327](#)
 - fasp_smat_identity_nc5, [327](#)
 - fasp_smat_identity_nc7, [327](#)
- smoother_bsr.c, [328](#)
 - fasp_smoother_dbsr_gs, [329](#)
 - fasp_smoother_dbsr_gs1, [329](#)
 - fasp_smoother_dbsr_gs_ascend, [330](#)
 - fasp_smoother_dbsr_gs_ascend1, [330](#)
 - fasp_smoother_dbsr_gs_descend, [331](#)
 - fasp_smoother_dbsr_gs_descend1, [331](#)
 - fasp_smoother_dbsr_gs_order1, [332](#)
 - fasp_smoother_dbsr_gs_order2, [332](#)
 - fasp_smoother_dbsr_ilu, [333](#)
 - fasp_smoother_dbsr_jacobi, [333](#)
 - fasp_smoother_dbsr_jacobi1, [333](#)
 - fasp_smoother_dbsr_jacobi_setup, [335](#)
 - fasp_smoother_dbsr_sor, [335](#)
 - fasp_smoother_dbsr_sor1, [336](#)
 - fasp_smoother_dbsr_sor_ascend, [336](#)
 - fasp_smoother_dbsr_sor_descend, [337](#)
 - fasp_smoother_dbsr_sor_order, [337](#)
- smoother_csr.c, [338](#)
 - fasp_smoother_dcsr_L1diag, [341](#)
 - fasp_smoother_dcsr_gs, [339](#)
 - fasp_smoother_dcsr_gs_cf, [339](#)
 - fasp_smoother_dcsr_ilu, [340](#)
 - fasp_smoother_dcsr_jacobi, [340](#)
 - fasp_smoother_dcsr_kaczmarz, [341](#)
 - fasp_smoother_dcsr_sgs, [342](#)
 - fasp_smoother_dcsr_sor, [342](#)
 - fasp_smoother_dcsr_sor_cf, [343](#)
- smoother_csr_cr.c, [343](#)
 - fasp_smoother_dcsr_gscr, [344](#)
- smoother_csr_poly.c, [345](#)
 - fasp_smoother_dcsr_poly, [345](#)
 - fasp_smoother_dcsr_poly_old, [345](#)
- smoother_str.c, [346](#)
 - fasp_generate_diaginv_block, [347](#)
 - fasp_smoother_dstr_gs, [347](#)
 - fasp_smoother_dstr_gs1, [348](#)
 - fasp_smoother_dstr_gs_ascend, [348](#)
 - fasp_smoother_dstr_gs_cf, [349](#)
 - fasp_smoother_dstr_gs_descend, [349](#)
 - fasp_smoother_dstr_gs_order, [350](#)
 - fasp_smoother_dstr_jacobi, [350](#)
 - fasp_smoother_dstr_jacobi1, [351](#)
 - fasp_smoother_dstr_schwarz, [351](#)
 - fasp_smoother_dstr_sor, [351](#)
 - fasp_smoother_dstr_sor1, [352](#)
 - fasp_smoother_dstr_sor_ascend, [352](#)
 - fasp_smoother_dstr_sor_cf, [353](#)
 - fasp_smoother_dstr_sor_descend, [353](#)
 - fasp_smoother_dstr_sor_order, [354](#)
- solver_type
 - input_param, [44](#)
- sparse_block.c, [354](#)
 - fasp_dbsr_getblk, [355](#)
 - fasp_dbsr_getblk_dcsr, [355](#)
 - fasp_dcsr_getblk, [356](#)

- sparse_bsr.c, 356
 - fasp_dbsr_alloc, 357
 - fasp_dbsr_cp, 358
 - fasp_dbsr_create, 358
 - fasp_dbsr_diaginv, 359
 - fasp_dbsr_diaginv2, 359
 - fasp_dbsr_diaginv3, 359
 - fasp_dbsr_diaginv4, 360
 - fasp_dbsr_diagpref, 360
 - fasp_dbsr_free, 361
 - fasp_dbsr_getdiag, 361
 - fasp_dbsr_getdiaginv, 362
 - fasp_dbsr_null, 362
 - fasp_dbsr_trans, 363
- sparse_coo.c, 363
 - fasp_dcoo_alloc, 364
 - fasp_dcoo_create, 364
 - fasp_dcoo_free, 364
 - fasp_dcoo_shift, 365
- sparse_csr.c, 365
 - fasp_dcsr_alloc, 367
 - fasp_dcsr_compress, 367
 - fasp_dcsr_compress_inplace, 367
 - fasp_dcsr_cp, 368
 - fasp_dcsr_create, 368
 - fasp_dcsr_diagpref, 369
 - fasp_dcsr_free, 369
 - fasp_dcsr_getcol, 370
 - fasp_dcsr_getdiag, 370
 - fasp_dcsr_multicoloring, 370
 - fasp_dcsr_null, 372
 - fasp_dcsr_perm, 372
 - fasp_dcsr_regdiag, 373
 - fasp_dcsr_shift, 373
 - fasp_dcsr_sort, 373
 - fasp_dcsr_symdiagscale, 375
 - fasp_dcsr_sympat, 375
 - fasp_dcsr_trans, 375
 - fasp_icsr_cp, 377
 - fasp_icsr_create, 377
 - fasp_icsr_free, 378
 - fasp_icsr_null, 378
 - fasp_icsr_trans, 378
- sparse_csrl.c, 379
 - fasp_dcsrl_create, 379
 - fasp_dcsrl_free, 380
- sparse_str.c, 380
 - fasp_dstr_alloc, 381
 - fasp_dstr_cp, 381
 - fasp_dstr_create, 382
 - fasp_dstr_free, 382
 - fasp_dstr_null, 382
- sparse_util.c, 383
 - fasp_sparse_MIS, 386
 - fasp_sparse_aat, 384
 - fasp_sparse_abyb, 384
 - fasp_sparse_abybms, 385
 - fasp_sparse_aplbms, 385
 - fasp_sparse_aplusb, 386
 - fasp_sparse_iit, 386
 - fasp_sparse_rapcmp, 387
 - fasp_sparse_rapms, 387
 - fasp_sparse_wta, 388
 - fasp_sparse_wtams, 388
 - fasp_sparse_ytx, 390
 - fasp_sparse_ytxbig, 390
- spbcgs.c, 391
 - fasp_solver_bdcsr_spbcgs, 392
 - fasp_solver_dbsr_spbcgs, 393
 - fasp_solver_dcsr_spbcgs, 393
 - fasp_solver_dstr_spbcgs, 395
- spcg.c, 396
 - fasp_solver_bdcsr_spcg, 397
 - fasp_solver_dcsr_spcg, 398
 - fasp_solver_dstr_spcg, 398
- spgmres.c, 399
 - fasp_solver_bdcsr_spgmres, 400
 - fasp_solver_dbsr_spgmres, 400
 - fasp_solver_dcsr_spgmres, 401
 - fasp_solver_dstr_spgmres, 401
- spminres.c, 402
 - fasp_solver_bdcsr_spminres, 404
 - fasp_solver_dcsr_spminres, 404
 - fasp_solver_dstr_spminres, 405
- spvgmres.c, 406
 - fasp_solver_bdcsr_spvgmres, 406
 - fasp_solver_dbsr_spvgmres, 407
 - fasp_solver_dcsr_spvgmres, 407
 - fasp_solver_dstr_spvgmres, 408
- stop_type
 - input_param, 44
 - itsolver_param, 45
- t
 - grid2d, 33
- THDs_AMG_GS
 - itsolver_bsr.c, 210
- THDs_CPR_gGS
 - itsolver_bsr.c, 210
- THDs_CPR_IGS
 - itsolver_bsr.c, 210
- TRUE
 - messages.h, 247
- tfather
 - grid2d, 34
- threads.c, 409
 - FASP_GET_START_END, 409
- timing.c, 410

- fasp_gettime, [410](#)
- tol
 - itsolver_param, [45](#)
 - precond_FASP_blkoi_data, [62](#)
- total_alloc_count
 - fasp.h, [151](#)
 - memory.c, [227](#)
- total_alloc_mem
 - fasp.h, [151](#)
 - memory.c, [227](#)
- triangles
 - grid2d, [34](#)
- UA_AMG
 - messages.h, [247](#)
- UNPT
 - messages.h, [247](#)
- USERDEFINED
 - messages.h, [247](#)
- V_CYCLE
 - messages.h, [247](#)
- val
 - dBSRmat, [27](#)
- vec.c, [411](#)
 - fasp_dvec_alloc, [412](#)
 - fasp_dvec_cp, [412](#)
 - fasp_dvec_create, [412](#)
 - fasp_dvec_free, [413](#)
 - fasp_dvec_isnan, [413](#)
 - fasp_dvec_maxdiff, [413](#)
 - fasp_dvec_null, [414](#)
 - fasp_dvec_rand, [414](#)
 - fasp_dvec_set, [415](#)
 - fasp_dvec_symdiagscale, [415](#)
 - fasp_ivec_alloc, [416](#)
 - fasp_ivec_create, [416](#)
 - fasp_ivec_free, [416](#)
 - fasp_ivec_set, [418](#)
- vertices
 - grid2d, [34](#)
- w
 - precond_block_reservoir_data, [52](#)
 - precond_FASP_blkoi_data, [62](#)
- W_CYCLE
 - messages.h, [247](#)
- WW
 - precond_block_reservoir_data, [52](#)
 - precond_FASP_blkoi_data, [62](#)
- workdir
 - input_param, [44](#)
- wrapper.c, [418](#)
 - fasp_fwrapper_amg_, [419](#)
 - fasp_fwrapper_krylov_amg_, [419](#)
- fasp_wrapper_dbsr_krylov_amg, [420](#)
- fasp_wrapper_dcoo_dbsr_krylov_amg, [420](#)