Fast Auxiliary Space Preconditioning

1.9.7 Jan/31/2017

Generated by Doxygen 1.8.13

Contents

Chapter 1

Introduction

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

We believe that there is no one-size-for-all solution method for discrete linear systemsfrom different applications. And, efficient iterative solvers can be constructed by taking the properties of PDEs and discretizations into account. In this project, we plan to construct a pool of discrete problems arising from partial differential equations (PDEs) or 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.

The structure of FASP is designed as follows:

- Level 0 (Aux∗.c): Auxiliary functions (timing, memory, threading, ...)
- Level 1 (Bla∗.c): Basic linear algebra subroutines (SpMV, RAP, ILU, SWZ, ...)
- Level 2 (Itr∗.c): Iterative methods and smoothers (Jacobi, GS, SOR, Poly, ...)
- Level 3 (Kry∗.c): Krylov iterative methods (CG, BiCGstab, MinRes, GMRES, ...)
- Level 4 (Pre∗.c): Preconditioners (GMG, AMG, FAMG, ...)
- Level 5 (Sol∗.c): User interface for FASP solvers (Solvers, wrappers, ...)
- Level x (Xtr∗.c): Interface to external packages (Mumps, Umfpack, ...)

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`.

Chapter 2

How to obtain FASP

The most updated version of FASP can be downloaded from

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

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

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

will give you the developer version of the FASP package.

Chapter 3

Building and Installation

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

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

```
$ make config
```

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

```
$ make install
```

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

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

```
$ wish fasp_install.tcl
```

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

```
$ make help
```

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

Chapter 4

Developers

Project leader:

- Xu, Jinchao (Penn State University, USA)

Project coordinator:

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

Current active developers (in alphabetic order):

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

With contributions from (in alphabetic order):

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

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

Chapter 5

Doxygen

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

`http://www.doxygen.org`

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

Chapter 6

Data Structure Index

6.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 7

File Index

7.1 File List

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

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 $<=$ max_levels
- dCSRmat A

  pointer to the matrix at level level_num
- dCSRmat R

  restriction operator at level level_num
- dCSRmat P

  prolongation operator at level level_num
- dvector b

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

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

  pointer to the numerical factorization from UMFPACK
- Pardiso_data pdata

  data for Intel MKL PARDISO
- ivector cfmark

  pointer to the CF marker at level level_num
- INT ILU_levels

  number of levels use ILU smoother

- ILU_data LU

  ILU matrix for ILU smoother.
- INT near_kernel_dim

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

  basis of near kernel space for SAMG
- INT SWZ_levels

  number of levels use Schwarz smoother
- SWZ_data Schwarz

  data of Schwarz smoother
- dvector w

  temporary work space
- Mumps_data mumps

  data for MUMPS
- INT cycle_type

  cycle type
- INT * ic

  indices for different colors
- INT * icmap

  mapping from vertex to color
- INT colors

  number of colors
- REAL weight

  weight for smoother

8.1.1 Detailed Description

Data for AMG solvers.

Note

This is needed for the AMG solver/preconditioner.

Definition at line 771 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 diaginv

  pointer to the diagonal inverse at level level_num
- dCSRmat Ac

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

  pointer to the numerical dactorization from UMFPACK
- Pardiso_data pdata

  data for Intel MKL PARDISO
- dCSRmat PP

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

  pointer to the auxiliary vectors for pressure block
- dBSRmat SS

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

  pointer to the auxiliary vectors for saturation block
- dvector diaginv_SS

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

  ILU data for pressure block.
- ivector cfmark

  pointer to the CF marker at level level_num
- INT ILU_levels

  number of levels use ILU smoother
- ILU_data LU

  ILU matrix for ILU smoother.
- INT near_kernel_dim

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

  basis of near kernel space for SAMG
- dCSRmat * A_nk

Matrix data for near kernal.
- dCSRmat ∗ P_nk

  Prolongation for near kernal.
- dCSRmat ∗ R_nk

  Resriction for near kernal.
- dvector w

  temporary work space
- Mumps_data mumps

  data for MUMPS

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 146 of file fasp_block.h.

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

- fasp_block.h

8.3 AMG_param Struct Reference

Parameters for AMG solver.

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

Data Fields

- SHORT AMG_type

  type of AMG method
- SHORT print_level

  print level for AMG
- INT maxit

  max number of iterations of AMG
- REAL tol

  stopping tolerance for AMG solver
- SHORT max_levels

  max number of levels of AMG
- INT coarse_dof

  max number of coarsest level DOF

- SHORT cycle_type

  type of AMG cycle
- REAL quality_bound

  quality threshold for pairwise aggregation
- SHORT smoother

  smoother type
- SHORT smooth_order

  smoother order
- SHORT presmooth_iter

  number of presmoothers
- SHORT postsmooth_iter

  number of postsmoothers
- REAL relaxation

  relaxation parameter for SOR smoother
- SHORT polynomial_degree

  degree of the polynomial smoother
- SHORT coarse_solver

  coarse solver type
- SHORT coarse_scaling

  switch of scaling of the coarse grid correction
- SHORT amli_degree

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

  coefficients of the polynomial used by AMLI cycle
- SHORT nl_amli_krylov_type

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

  coarsening type
- SHORT aggregation_type

  aggregation type
- SHORT interpolation_type

  interpolation type
- REAL strong_threshold

  strong connection threshold for coarsening
- REAL max_row_sum

  maximal row sum parameter
- REAL truncation_threshold

  truncation threshold
- INT aggressive_level

  number of levels use aggressive coarsening
- INT aggressive_path

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

  number of pairwise matchings
- REAL strong_coupled

  strong coupled threshold for aggregate
- INT max_aggregation

max size of each aggregate

- REAL tentative_smooth

  relaxation parameter for smoothing the tentative prolongation
- SHORT smooth_filter

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

  number of levels use ILU smoother
- SHORT ILU_type

  ILU type for smoothing.
- INT ILU_lfil

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

  drop tolerance for ILUt
- REAL ILU_relax

  relaxation for ILUs
- REAL ILU_permtol

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

  number of levels use Schwarz smoother
- INT SWZ_mmsize

  maximal block size
- INT SWZ_maxlvl

  maximal levels
- INT SWZ_type

  type of Schwarz method
- INT SWZ_blksolver

  type of Schwarz block solver

8.3.1 Detailed Description

Parameters for AMG solver.

Note

This is needed for the AMG solver/preconditioner.

Definition at line 442 of file fasp.h.

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

- fasp.h

8.4 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]

8.4.1 Detailed Description

Block REAL vector structure.

Definition at line 110 of file fasp_block.h.

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

- fasp_block.h

8.5 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]

8.5.1 Detailed Description

Block INT vector structure.

Note

The starting index of A is 0.

Definition at line 126 of file fasp_block.h.

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

- fasp_block.h

8.6 dBLCmat Struct Reference

Block REAL CSR matrix format.

`#include <fasp_block.h>`

Data Fields

- INT brow

    row number of blocks in A, m
- INT bcol

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

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

8.6.1 Detailed Description

Block REAL CSR matrix format.

Note

The starting index of A is 0.

Definition at line 74 of file fasp_block.h.

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

- fasp_block.h

8.7 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

8.7.1 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 34 of file fasp_block.h.

8.7.2 Field Documentation

8.7.2.1 JA

```
INT* JA
```

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

Definition at line 64 of file fasp_block.h.

8.7.2.2 val

```
REAL* val
```

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

Definition at line 57 of file fasp_block.h.

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

- fasp_block.h

8.8 dCOOmat Struct Reference

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

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

Data Fields

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

8.8.1 Detailed Description

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

Coordinate Format (I,J,A)

Note

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

Definition at line 208 of file fasp.h.

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

- fasp.h

8.9 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

8.9.1 Detailed Description

Sparse matrix of REAL type in CSRL format.

Definition at line 264 of file fasp.h.

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

- fasp.h

8.10 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

8.10.1 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 147 of file fasp.h.

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

- fasp.h

8.11 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

8.11.1 Detailed Description

Dense matrix of REAL type.

A dense REAL matrix

Definition at line 107 of file fasp.h.

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

- fasp.h

8.12 dSTRmat Struct Reference

Structure matrix of REAL type.

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

Data Fields

- INT nx
  number of grids in x direction
- INT ny
  number of grids in y direction
- INT nz
  number of grids in z direction
- INT nxy
  number of grids on x-y plane
- INT nc
  size of each block (number of components)
- INT ngrid
  number of grids
- REAL ∗ diag
  diagonal entries (length is ngrid∗(nc^2))
- INT nband
  number of off-diag bands
- INT ∗ offsets
  offsets of the off-diagonals (length is nband)
- REAL ∗∗ offdiag
  off-diagonal entries (dimension is nband ∗ [(ngrid-|offsets|) ∗ nc^2])

8.12.1 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 303 of file fasp.h.

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

- fasp.h

8.13 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

8.13.1 Detailed Description

Vector with n entries of REAL type.

Definition at line 341 of file fasp.h.

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

- fasp.h

8.14 grid2d Struct Reference

Two dimensional grid data structure.

```
#include <fasp_grid.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

8.14.1 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 24 of file fasp_grid.h.

8.14.2 Field Documentation

8.14.2.1 e

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

Vertices of edges

Definition at line 27 of file fasp_grid.h.

8.14.2.2 edges

```
INT edges
```

Number of edges

Definition at line 38 of file fasp_grid.h.

8.14.2.3 ediri

```
INT* ediri
```

Boundary flags (0 <=> interior edge)

Definition at line 31 of file fasp_grid.h.

8.14.2.4 efather

```
INT* efather
```

Father edge or triangle

Definition at line 34 of file fasp_grid.h.

8.14.2.5 p

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

Coordinates of vertices

Definition at line 26 of file fasp_grid.h.

8.14.2.6 pdiri

```
INT* pdiri
```

Boundary flags (0 <=> interior point)

Definition at line 30 of file fasp_grid.h.

8.14.2.7 pfather

```
INT* pfather
```

Father point or edge

Definition at line 33 of file fasp_grid.h.

8.14.2.8 s

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

Edges of triangles

Definition at line 29 of file fasp_grid.h.

8.14.2.9 t

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

Vertices of triangles

Definition at line 28 of file fasp_grid.h.

8.14.2.10 tfather

```
INT* tfather
```

Father triangle

Definition at line 35 of file fasp_grid.h.

8.14.2.11 triangles

```
INT triangles
```

Number of triangles

Definition at line 39 of file fasp_grid.h.

8.14.2.12 vertices

```
INT vertices
```

Number of grid points

Definition at line 37 of file fasp_grid.h.

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

- fasp_grid.h

8.15 iBLCmat 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]

8.15.1 Detailed Description

Block INT CSR matrix format.

Note

The starting index of A is 0.

Definition at line 93 of file fasp_block.h.

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

- fasp_block.h

8.16 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

8.16.1 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 238 of file fasp.h.

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

- fasp.h

8.17 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

8.17.1 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 177 of file fasp.h.

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

- fasp.h

8.18 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

8.18.1 Detailed Description

Dense matrix of INT type.

A dense INT matrix

Definition at line 126 of file fasp.h.

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

- fasp.h

8.19 ILU_data Struct Reference

Data for ILU setup.

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

Data Fields

- INT row

  row number of matrix LU, m
- INT col

  column of matrix LU, n
- INT nzlu

  number of nonzero entries
- INT ∗ ijlu

  integer array of row pointers and column indexes, the size is nzlu
- REAL ∗ luval

  nonzero entries of LU
- INT nb

  block size for BSR type only
- INT nwork

  work space size
- REAL ∗ work

  work space
- INT ncolors

  number of colors for multi-threading
- INT ∗ ic

  indices for different colors
- INT ∗ icmap

  mapping from vertex to color
- INT ∗ uptr

  temporary work space
- INT nlevL

  number of colors for lower triangle
- INT nlevU

  number of colors for upper triangle
- INT ∗ ilevL

  number of vertices in each color for lower triangle
- INT ∗ ilevU

  number of vertices in each color for upper triangle
- INT ∗ jlevL

  mapping from row to color for lower triangle
- INT ∗ jlevU

  mapping from row to color for upper triangle

8.19.1 Detailed Description

Data for ILU setup.

Definition at line 707 of file fasp.h.

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

- fasp.h

8.20 ILU_param Struct Reference

Parameters for ILU.

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

Data Fields

- SHORT print_level
  print level
- SHORT ILU_type
  ILU type for decomposition.
- INT ILU_lfil
  level of fill-in for ILUk
- REAL ILU_droptol
  drop tolerance for ILUt
- REAL ILU_relax
  add the sum of dropped elements to diagonal element in proportion relax
- REAL ILU_permtol
  permuted if permtol$*|a(i,j)| > |a(i,i)|$

8.20.1 Detailed Description

Parameters for ILU.

Definition at line 389 of file fasp.h.

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

- fasp.h

8.21 input_param Struct Reference

Input parameters.

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

Data Fields

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

8.21.1 Detailed Description

Input parameters.

Input parameters, reading from disk file

Definition at line 1087 of file fasp.h.

8.21.2 Field Documentation

8.21.2.1 AMG_aggregation_type

`SHORT AMG_aggregation_type`

aggregation type

Definition at line 1141 of file fasp.h.

8.21.2.2 AMG_aggressive_level

`INT AMG_aggressive_level`

number of levels use aggressive coarsening

Definition at line 1146 of file fasp.h.

8.21.2.3 AMG_aggressive_path

`INT AMG_aggressive_path`

number of paths used to determine strongly coupled C-set

Definition at line 1147 of file fasp.h.

8.21.2.4 AMG_amli_degree

```
SHORT AMG_amli_degree
```

degree of the polynomial used by AMLI cycle

Definition at line 1135 of file fasp.h.

8.21.2.5 AMG_coarse_dof

```
INT AMG_coarse_dof
```

max number of coarsest level DOF

Definition at line 1129 of file fasp.h.

8.21.2.6 AMG_coarse_scaling

```
SHORT AMG_coarse_scaling
```

switch of scaling of the coarse grid correction

Definition at line 1134 of file fasp.h.

8.21.2.7 AMG_coarse_solver

```
SHORT AMG_coarse_solver
```

coarse solver type

Definition at line 1133 of file fasp.h.

8.21.2.8 AMG_coarsening_type

```
SHORT AMG_coarsening_type
```

coarsening type

Definition at line 1140 of file fasp.h.

8.21.2.9 AMG_cycle_type

`SHORT AMG_cycle_type`

type of cycle

Definition at line 1122 of file fasp.h.

8.21.2.10 AMG_ILU_levels

`SHORT AMG_ILU_levels`

how many levels use ILU smoother

Definition at line 1132 of file fasp.h.

8.21.2.11 AMG_interpolation_type

`SHORT AMG_interpolation_type`

interpolation type

Definition at line 1142 of file fasp.h.

8.21.2.12 AMG_levels

`SHORT AMG_levels`

maximal number of levels

Definition at line 1121 of file fasp.h.

8.21.2.13 AMG_max_aggregation

`INT AMG_max_aggregation`

max size of each aggregate

Definition at line 1153 of file fasp.h.

8.21.2.14 AMG_max_row_sum

```
REAL AMG_max_row_sum
```

maximal row sum

Definition at line 1145 of file fasp.h.

8.21.2.15 AMG_maxit

```
INT AMG_maxit
```

number of iterations for AMG used as preconditioner

Definition at line 1131 of file fasp.h.

8.21.2.16 AMG_nl_amli_krylov_type

```
SHORT AMG_nl_amli_krylov_type
```

type of Krylov method used by nonlinear AMLI cycle

Definition at line 1136 of file fasp.h.

8.21.2.17 AMG_pair_number

```
INT AMG_pair_number
```

number of pairs in matching algorithm

Definition at line 1148 of file fasp.h.

8.21.2.18 AMG_polynomial_degree

```
SHORT AMG_polynomial_degree
```

degree of the polynomial smoother

Definition at line 1126 of file fasp.h.

8.21.2.19 AMG_postsmooth_iter

```
SHORT AMG_postsmooth_iter
```

number of postsmoothing

Definition at line 1128 of file fasp.h.

8.21.2.20 AMG_presmooth_iter

```
SHORT AMG_presmooth_iter
```

number of presmoothing

Definition at line 1127 of file fasp.h.

8.21.2.21 AMG_quality_bound

```
REAL AMG_quality_bound
```

threshold for pair wise aggregation

Definition at line 1149 of file fasp.h.

8.21.2.22 AMG_relaxation

```
REAL AMG_relaxation
```

over-relaxation parameter for SOR

Definition at line 1125 of file fasp.h.

8.21.2.23 AMG_smooth_filter

```
SHORT AMG_smooth_filter
```

use filter for smoothing the tentative prolongation or not

Definition at line 1155 of file fasp.h.

8.21.2.24 AMG_smooth_order

```
SHORT AMG_smooth_order
```

order for smoothers

Definition at line 1124 of file fasp.h.

8.21.2.25 AMG_smoother

```
SHORT AMG_smoother
```

type of smoother

Definition at line 1123 of file fasp.h.

8.21.2.26 AMG_strong_coupled

```
REAL AMG_strong_coupled
```

strong coupled threshold for aggregate

Definition at line 1152 of file fasp.h.

8.21.2.27 AMG_strong_threshold

```
REAL AMG_strong_threshold
```

strong threshold for coarsening

Definition at line 1143 of file fasp.h.

8.21.2.28 AMG_SWZ_levels

```
INT AMG_SWZ_levels
```

number of levels use Schwarz smoother

Definition at line 1137 of file fasp.h.

8.21.2.29 AMG_tentative_smooth

`REAL AMG_tentative_smooth`

relaxation factor for smoothing the tentative prolongation

Definition at line 1154 of file fasp.h.

8.21.2.30 AMG_tol

`REAL AMG_tol`

tolerance for AMG if used as preconditioner

Definition at line 1130 of file fasp.h.

8.21.2.31 AMG_truncation_threshold

`REAL AMG_truncation_threshold`

truncation factor for interpolation

Definition at line 1144 of file fasp.h.

8.21.2.32 AMG_type

`SHORT AMG_type`

Type of AMG

Definition at line 1120 of file fasp.h.

8.21.2.33 ILU_droptol

`REAL ILU_droptol`

drop tolerance

Definition at line 1109 of file fasp.h.

8.21.2.34 ILU_lfil

```
INT ILU_lfil
```

level of fill-in

Definition at line 1108 of file fasp.h.

8.21.2.35 ILU_permtol

```
REAL ILU_permtol
```

permutation tolerance

Definition at line 1111 of file fasp.h.

8.21.2.36 ILU_relax

```
REAL ILU_relax
```

scaling factor: add the sum of dropped entries to diagonal

Definition at line 1110 of file fasp.h.

8.21.2.37 ILU_type

```
SHORT ILU_type
```

ILU type for decomposition

Definition at line 1107 of file fasp.h.

8.21.2.38 inifile

```
char inifile[256]
```

ini file name

Definition at line 1094 of file fasp.h.

8.21.2.39 itsolver_maxit

```
INT itsolver_maxit
```

maximal number of iterations for iterative solvers

Definition at line 1103 of file fasp.h.

8.21.2.40 itsolver_tol

```
REAL itsolver_tol
```

tolerance for iterative linear solver

Definition at line 1102 of file fasp.h.

8.21.2.41 output_type

```
SHORT output_type
```

type of output stream

Definition at line 1091 of file fasp.h.

8.21.2.42 precond_type

```
SHORT precond_type
```

type of preconditioner for iterative solvers

Definition at line 1100 of file fasp.h.

8.21.2.43 print_level

```
SHORT print_level
```

print level

Definition at line 1090 of file fasp.h.

8.21.2.44 problem_num

`INT problem_num`

problem number to solve

Definition at line 1096 of file fasp.h.

8.21.2.45 restart

`INT restart`

restart number used in GMRES

Definition at line 1104 of file fasp.h.

8.21.2.46 solver_type

`SHORT solver_type`

type of iterative solvers

Definition at line 1099 of file fasp.h.

8.21.2.47 stop_type

`SHORT stop_type`

type of stopping criteria for iterative solvers

Definition at line 1101 of file fasp.h.

8.21.2.48 SWZ_blksolver

`INT SWZ_blksolver`

type of Schwarz block solver

Definition at line 1117 of file fasp.h.

8.21.2.49 SWZ_maxlvl

```
INT SWZ_maxlvl
```

maximal levels

Definition at line 1115 of file fasp.h.

8.21.2.50 SWZ_mmsize

```
INT SWZ_mmsize
```

maximal block size

Definition at line 1114 of file fasp.h.

8.21.2.51 SWZ_type

```
INT SWZ_type
```

type of Schwarz method

Definition at line 1116 of file fasp.h.

8.21.2.52 workdir

```
char workdir[256]
```

working directory for data files

Definition at line 1095 of file fasp.h.

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

- fasp.h

8.22 ITS_param Struct Reference

Parameters passed to iterative solvers.

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

Data Fields

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

8.22.1 Detailed Description

Parameters passed to iterative solvers.

Definition at line 373 of file fasp.h.

8.22.2 Field Documentation

8.22.2.1 itsolver_type

`SHORT itsolver_type`

solver type: see message.h

Definition at line 375 of file fasp.h.

8.22.2.2 maxit

`INT maxit`

max number of iterations

Definition at line 378 of file fasp.h.

8.22.2.3 precond_type

`SHORT precond_type`

preconditioner type: see message.h

Definition at line 376 of file fasp.h.

8.22.2.4 print_level

`SHORT print_level`

print level: 0–10

Definition at line 381 of file fasp.h.

8.22.2.5 restart

`INT restart`

number of steps for restarting: for GMRES etc

Definition at line 380 of file fasp.h.

8.22.2.6 stop_type

`SHORT stop_type`

stopping criteria type

Definition at line 377 of file fasp.h.

8.22.2.7 tol

`REAL tol`

convergence tolerance

Definition at line 379 of file fasp.h.

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

- fasp.h

8.23 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

8.23.1 Detailed Description

Vector with n entries of INT type.

Definition at line 355 of file fasp.h.

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

- fasp.h

8.24 Mumps_data Struct Reference

Parameters for MUMPS interface.

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

Data Fields

- INT job
  work for MUMPS

8.24.1 Detailed Description

Parameters for MUMPS interface.

Added on 10/10/2014

Definition at line 585 of file fasp.h.

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

- fasp.h

8.25 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)(const void ∗, const REAL ∗, REAL ∗)
  action for MxV, void function pointer

8.25.1 Detailed Description

Matrix-vector multiplication, replace the actual matrix.

Definition at line 1071 of file fasp.h.

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

- fasp.h

8.26 Pardiso_data Struct Reference

Parameters for Intel MKL PARDISO interface.

`#include <fasp.h>`

Data Fields

- void ∗ pt [64]
  Internal solver memory pointer.

8.26.1 Detailed Description

Parameters for Intel MKL PARDISO interface.

Added on 11/28/2015

Definition at line 603 of file fasp.h.

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

- fasp.h

8.27 precond Struct Reference

Preconditioner data and action.

`#include <fasp.h>`

Data Fields

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

8.27.1 Detailed Description

Preconditioner data and action.

Note

This is the preconditioner structure for preconditioned iterative methods.

Definition at line 1057 of file fasp.h.

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

- fasp.h

8.28 precond_block_data Struct Reference

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

`#include <fasp_block.h>`

Data Fields

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

8.28.1 Detailed Description

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

This is needed for the block preconditioner.

Definition at line 350 of file fasp_block.h.

8.28.2 Field Documentation

8.28.2.1 A_diag

`dCSRmat* A_diag`

data for each diagonal block

Definition at line 357 of file fasp_block.h.

8.28.2.2 Ablc

`dBLCmat* Ablc`

problem data, the blocks

Definition at line 355 of file fasp_block.h.

8.28.2.3 amgparam

`AMG_param* amgparam`

parameters for AMG

Definition at line 369 of file fasp_block.h.

8.28.2.4 LU_diag

```
void** LU_diag
```

LU decomposition for the diagonal blocks (for UMFpack)

Definition at line 365 of file fasp_block.h.

8.28.2.5 mgl

```
AMG_data** mgl
```

AMG data for the diagonal blocks

Definition at line 368 of file fasp_block.h.

8.28.2.6 r

```
dvector r
```

temp work space

Definition at line 359 of file fasp_block.h.

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

- fasp_block.h

8.29 precond_data Struct Reference

Data passed to the preconditioners.

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

Data Fields

- SHORT AMG_type

  type of AMG method
- SHORT print_level

  print level in AMG preconditioner
- INT maxit

  max number of iterations of AMG preconditioner
- SHORT max_levels

  max number of AMG levels
- REAL tol

  tolerance for AMG preconditioner
- SHORT cycle_type

  AMG cycle type.
- SHORT smoother

  AMG smoother type.
- SHORT smooth_order

  AMG smoother ordering.
- SHORT presmooth_iter

  number of presmoothing
- SHORT postsmooth_iter

  number of postsmoothing
- REAL relaxation

  relaxation parameter for SOR smoother
- SHORT polynomial_degree

  degree of the polynomial smoother
- SHORT coarsening_type

  switch of scaling of the coarse grid correction
- SHORT coarse_solver

  coarse solver type for AMG
- SHORT coarse_scaling

  switch of scaling of the coarse grid correction
- SHORT amli_degree

  degree of the polynomial used by AMLI cycle
- SHORT nl_amli_krylov_type

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

  smooth factor for smoothing the tentative prolongation
- REAL ∗ amli_coef

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

  AMG preconditioner data.
- ILU_data ∗ LU

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

  Matrix data.
- dCSRmat ∗ A_nk

Matrix data for near kernel.

- dCSRmat ∗ P_nk

  Prolongation for near kernel.
- dCSRmat ∗ R_nk

  Restriction for near kernel.
- dvector r

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

  temporary work space for other usage

8.29.1 Detailed Description

Data passed to the preconditioners.

Definition at line 856 of file fasp.h.

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

- fasp.h

8.30 precond_data_bsr Struct Reference

Data passed to the preconditioners.

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

Data Fields

- SHORT AMG_type

  type of AMG method
- SHORT print_level

  print level in AMG preconditioner
- INT maxit

  max number of iterations of AMG preconditioner
- INT max_levels

  max number of AMG levels
- REAL tol

  tolerance for AMG preconditioner
- SHORT cycle_type

  AMG cycle type.
- SHORT smoother

  AMG smoother type.
- SHORT smooth_order

  AMG smoother ordering.

- SHORT presmooth_iter
  number of presmoothing
- SHORT postsmooth_iter
  number of postsmoothing
- SHORT coarsening_type
  coarsening type
- REAL relaxation
  relaxation parameter for SOR smoother
- SHORT coarse_solver
  coarse solver type for AMG
- SHORT coarse_scaling
  switch of scaling of the coarse grid correction
- SHORT amli_degree
  degree of the polynomial used by AMLI cycle
- REAL * amli_coef
  coefficients of the polynomial used by AMLI cycle
- REAL tentative_smooth
  smooth factor for smoothing the tentative prolongation
- SHORT nl_amli_krylov_type
  type of krylov method used by Nonlinear AMLI cycle
- AMG_data_bsr * mgl_data
  AMG preconditioner data.
- AMG_data * pres_mgl_data
  AMG preconditioner data for pressure block.
- ILU_data * LU
  ILU preconditioner data (needed for CPR type preconditioner)
- dBSRmat * A
  Matrix data.
- dCSRmat * A_nk
  Matrix data for near kernal.
- dCSRmat * P_nk
  Prolongation for near kernal.
- dCSRmat * R_nk
  Resriction for near kernal.
- dvector r
  temporary dvector used to store and restore the residual
- REAL * w
  temporary work space for other usage

8.30.1 Detailed Description

Data passed to the preconditioners.

Note

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

Definition at line 258 of file fasp_block.h.

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

- fasp_block.h

8.31 precond_data_str Struct Reference

Data passed to the preconditioner for dSTRmat matrices.

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

Data Fields

- SHORT AMG_type

    type of AMG method
- SHORT print_level

    print level in AMG preconditioner
- INT maxit

    max number of iterations of AMG preconditioner
- SHORT max_levels

    max number of AMG levels
- REAL tol

    tolerance for AMG preconditioner
- SHORT cycle_type

    AMG cycle type.
- SHORT smoother

    AMG smoother type.
- SHORT presmooth_iter

    number of presmoothing
- SHORT postsmooth_iter

    number of postsmoothing
- SHORT coarsening_type

    coarsening type
- REAL relaxation

    relaxation parameter for SOR smoother
- SHORT coarse_scaling

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

    AMG preconditioner data.
- ILU_data ∗ LU

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

    whether the matrix are scaled or not
- dCSRmat ∗ A

    the original CSR matrix
- dSTRmat ∗ A_str

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

    store Saturation block in STR format
- dvector ∗ diaginv

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

- ivector ∗ pivot

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

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

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

    order for smoothing
- ivector ∗ neigh

    array to store neighbor information
- dvector r

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

    temporary work space for other usage

8.31.1 Detailed Description

Data passed to the preconditioner for dSTRmat matrices.

Definition at line 949 of file fasp.h.

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

- fasp.h

8.32 precond_diag_str Struct Reference

Data passed to diagonal preconditioner for dSTRmat matrices.

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

Data Fields

- INT nc

    number of components
- dvector diag

    diagonal elements

8.32.1 Detailed Description

Data passed to diagonal preconditioner for dSTRmat matrices.

Note

This is needed for the diagonal preconditioner.

Definition at line 1041 of file fasp.h.

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

- fasp.h

8.33 precond_diagbsr Struct Reference

Data passed to diagnal preconditioner for dBSRmat matrices.

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

Data Fields

- INT nb

  dimension of each sub-block
- dvector diag

  diagnal elements

8.33.1 Detailed Description

Data passed to diagnal preconditioner for dBSRmat matrices.

Note

This is needed for the diagnal preconditioner.

Definition at line 241 of file fasp_block.h.

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

- fasp_block.h

8.34 precond_sweeping_data Struct Reference

Data passed to the preconditioner for sweeping preconditioning.

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

Data Fields

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

8.34.1 Detailed Description

Data passed to the preconditioner for sweeping preconditioning.

Author

Xiaozhe Hu

Date

05/01/2014

Note

This is needed for the sweeping preconditioner.

Definition at line 382 of file fasp_block.h.

8.34.2 Field Documentation

8.34.2.1 A

```
dBLCmat* A
```

problem data, the sparse matrix

Definition at line 386 of file fasp_block.h.

8.34.2.2 Ai

`dBLCmat* Ai`

preconditioner data, the sparse matrix

Definition at line 387 of file fasp_block.h.

8.34.2.3 local_A

`dCSRmat* local_A`

local stiffness matrix for each layer

Definition at line 389 of file fasp_block.h.

8.34.2.4 local_index

`ivector* local_index`

local index for each layer

Definition at line 392 of file fasp_block.h.

8.34.2.5 local_LU

`void** local_LU`

lcoal LU decomposition (for UMFpack)

Definition at line 390 of file fasp_block.h.

8.34.2.6 NumLayers

`INT NumLayers`

number of layers

Definition at line 384 of file fasp_block.h.

8.34.2.7 r

```
dvector r
```

temporary dvector used to store and restore the residual

Definition at line 395 of file fasp_block.h.

8.34.2.8 w

```
REAL* w
```

temporary work space for other usage

Definition at line 396 of file fasp_block.h.

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

- fasp_block.h

8.35 SWZ_data Struct Reference

Data for Schwarz methods.

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

Data Fields

- dCSRmat A
  pointer to the matrix
- INT nblk
  number of blocks
- INT * iblock
  row index of blocks
- INT * jblock
  column index of blocks
- REAL * rhsloc
  temp work space???
- dvector rhsloc1
  local right hand side
- dvector xloc1
  local solution
- REAL * au

LU decomposition: the U block.

- REAL ∗ al

  LU decomposition: the L block.
- INT SWZ_type

  Schwarz method type.
- INT blk_solver

  Schwarz block solver.
- INT memt

  working space size
- INT ∗ mask

  mask
- INT maxbs

  maximal block size
- INT ∗ maxa

  maxa
- dCSRmat ∗ blk_data

  matrix for each partition
- Mumps_data ∗ mumps

  param for MUMPS
- SWZ_param ∗ swzparam

  param for Schwarz

8.35.1 Detailed Description

Data for Schwarz methods.

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

Definition at line 631 of file fasp.h.

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

- fasp.h

8.36 SWZ_param Struct Reference

Parameters for Schwarz method.

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

Data Fields

- SHORT print_level
  print leve
- SHORT SWZ_type
  type for Schwarz method
- INT SWZ_maxlvl
  maximal level for constructing the blocks
- INT SWZ_mmsize
  maximal size of blocks
- INT SWZ_blksolver
  type of Schwarz block solver

8.36.1 Detailed Description

Parameters for Schwarz method.

Added on 05/14/2012

Definition at line 417 of file fasp.h.

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

- fasp.h

Chapter 9

File Documentation

9.1 AuxArray.c File Reference

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

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

Functions

- void fasp_darray_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_darray_cp (const INT n, const REAL ∗x, REAL ∗y)

  Copy an array to the other y=x.
- void fasp_iarray_cp (const INT n, const INT ∗x, INT ∗y)

  Copy an array to the other y=x.

9.1.1 Detailed Description

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

Note

This file contains Level-0 (Aux) functions. It requires: AuxThreads.c

9.1.2 Function Documentation

9.1.2.1 fasp_darray_cp()

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

Copy an array to the other y=x.

Parameters

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

Author

Chensong Zhang

Date

2010/04/03

Definition at line 164 of file AuxArray.c.

9.1.2.2 fasp_darray_set()

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

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

Parameters

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

Author

Chensong Zhang

Date

04/03/2010

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

Definition at line 41 of file AuxArray.c.

9.1.2.3 fasp_iarray_cp()

```
void fasp_iarray_cp (
            const INT n,
            const 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 184 of file AuxArray.c.

9.1.2.4 fasp_iarray_set()

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

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

Parameters

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

Author

Chensong Zhang

Date

04/03/2010

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

Definition at line 103 of file AuxArray.c.

9.2 AuxConvert.c File Reference

Utilities for encoding format conversion.

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

Functions

- unsigned long fasp_aux_change_endian4 (const unsigned long x)

  Swap order for different endian systems.
- double fasp_aux_change_endian8 (const double x)

  Swap order for different endian systems.
- double fasp_aux_bbyteToldouble (const unsigned char bytes[ ])

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

9.2.1 Detailed Description

Utilities for encoding format conversion.

Note

This file contains Level-0 (Aux) functions.

9.2.2 Function Documentation

9.2.2.1 fasp_aux_bbyteToldouble()

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

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

Parameters

bytes	A unsigned char

Returns

Unsigend long ineger after swapping

Author

Chensong Zhang

Date

11/16/2009

Definition at line 81 of file AuxConvert.c.

9.2.2.2 fasp_aux_change_endian4()

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

Swap order for different endian systems.

Parameters

x	An unsigned long integer

Returns

Unsigend long ineger after swapping

Author

Chensong Zhang

Date

11/16/2009

Definition at line 32 of file AuxConvert.c.

9.2.2.3 fasp_aux_change_endian8()

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

Swap order for different endian systems.

Parameters

x	A unsigned long integer

Returns

Unsigend long ineger after swapping

Author

Chensong Zhang

Date

11/16/2009

Definition at line 50 of file AuxConvert.c.

9.3 AuxGivens.c File Reference

Givens transformation.

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

Functions

- void fasp_aux_givens (const REAL beta, const dCSRmat ∗H, dvector ∗y, REAL ∗work)

  Perform Givens rotations to compute y |beta∗e_1- H∗y|.

9.3.1 Detailed Description

Givens transformation.

Note

This file contains Level-0 (Aux) functions.

9.3.2 Function Documentation

9.3.2.1 fasp_aux_givens()

```
void fasp_aux_givens (
            const REAL beta,
            const dCSRmat * H,
            dvector * y,
            REAL * work )
```

Perform Givens rotations to compute y |beta∗e_1- H∗y|.

Parameters

beta	Norm of residual r_0
H	Upper Hessenberg dCSRmat matrix: (m+1)∗m
y	Minimizer of |beta∗e_1- H∗y|
work	Temporary work array

Author

Xuehai Huang

Date

10/19/2008

Definition at line 35 of file AuxGivens.c.

9.4 AuxGraphics.c File Reference

Graphical output for CSR matrix.

```
#include <math.h>
#include "fasp.h"
#include "fasp_grid.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.
- INT fasp_dcsr_plot (const dCSRmat *A, const char *fname)

  Write dCSR sparse matrix pattern in BMP file format.
- void fasp_dbsr_subplot (const dBSRmat *A, const char *filename, INT size)

  Write sparse matrix pattern in BMP file format.
- INT fasp_dbsr_plot (const dBSRmat *A, const char *fname)

  Write dBSR sparse matrix pattern in BMP file format.
- void fasp_grid2d_plot (pgrid2d pg, INT level)

  Output grid to a EPS file.

9.4.1 Detailed Description

Graphical output for CSR matrix.

Note

This file contains Level-0 (Aux) functions. It requires: AuxMemory.c

9.4.2 Function Documentation

9.4.2.1 fasp_dbsr_plot()

```
void fasp_dbsr_plot (
            const dBSRmat * A,
            const char * fname )
```

Write dBSR sparse matrix pattern in BMP file format.

Parameters

A	Pointer to the dBSRmat matrix
fname	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 346 of file AuxGraphics.c.

9.4.2.2 fasp_dbsr_subplot()

```
void fasp_dbsr_subplot (
            const dBSRmat * A,
            const char * filename,
            INT size )
```

Write sparse matrix pattern in BMP file format.

Parameters

A	Pointer to the dBSRmat matrix
filename	File name
size	size∗size is the picture size for the picture

Author

Chunsheng Feng

Date

11/16/2013

Note

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

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

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

Definition at line 266 of file AuxGraphics.c.

9.4.2.3 fasp_dcsr_plot()

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

Write dCSR sparse matrix pattern in BMP file format.

Parameters

A	Pointer to the dBSRmat matrix
fname	File name to plot to

Author

Chunsheng Feng

Date

11/16/2013

Note

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

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

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

Definition at line 117 of file AuxGraphics.c.

9.4.2.4 fasp_dcsr_subplot()

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

Write sparse matrix pattern in BMP file format.

Parameters

A	Pointer to the dCSRmat matrix
filename	File name
size	size*size is the picture size for the picture

Author

Chensong Zhang

Date

03/29/2009

Note

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

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

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

Definition at line 57 of file AuxGraphics.c.

9.4.2.5 fasp_grid2d_plot()

```
void fasp_grid2d_plot (
            pgrid2d pg,
            INT level )
```

Output grid to a EPS file.

Parameters

pg	Pointer to grid in 2d
level	Number of levels

Author

Chensong Zhang

Date

03/29/2009

Definition at line 493 of file AuxGraphics.c.

9.5 AuxInput.c File Reference

Read and check input parameters.

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

Functions

- SHORT fasp_param_check (input_param *inparam)
  Simple check on input parameters.
- void fasp_param_input (const char *fname, input_param *inparam)
  Read input parameters from disk file.

9.5.1 Detailed Description

Read and check input parameters.

Note

This file contains Level-0 (Aux) functions. It requires: AuxMemory.c and AuxMessage.c

9.5.2 Function Documentation

9.5.2.1 fasp_param_check()

```
SHORT fasp_param_check (
            input_param * inparam )
```

Simple check on input parameters.

Parameters

inparam	Input parameters

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Chensong Zhang

Date

09/29/2013

Definition at line 33 of file AuxInput.c.

9.5.2.2 fasp_param_input()

```
void fasp_param_input (
            const char * fname,
            input_param * inparam )
```

Read input parameters from disk file.

Parameters

fname	File name for input file
inparam	Input parameters

Author

Chensong Zhang

Date

03/20/2010

Modified by Xiaozhe Hu on 01/23/2011: add AMLI cycle Modified by Chensong Zhang on 05/10/2013: add a new input. Modified by Chensong Zhang on 03/23/2015: skip unknown keyword.

Definition at line 108 of file AuxInput.c.

9.6 AuxMemory.c File Reference

Memory allocation and deallocation subroutines.

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

Functions

- void ∗ fasp_mem_calloc (const LONGLONG size, const INT type)

  1M = 1024∗1024
- void ∗ fasp_mem_realloc (void ∗oldmem, const LONGLONG tsize)

  Reallocate, initiate, and check memory.
- void fasp_mem_free (void ∗mem)

  Free up previous allocated memory body.
- void fasp_mem_usage ()

  Show total allocated memory currently.
- SHORT fasp_mem_iludata_check (const ILU_data ∗iludata)

  Check wether a ILU_data has enough work space.

Variables

- unsigned INT total_alloc_mem = 0
- unsigned INT total_alloc_count = 0

  Total allocated memory amount.
- const INT Million = 1048576

  Total number of allocations.

9.6.1 Detailed Description

Memory allocation and deallocation subroutines.

Note

This file contains Level-0 (Aux) functions.

9.6.2 Function Documentation

9.6.2.1 fasp_mem_calloc()

```
void * fasp_mem_calloc (
            const LONGLONG size,
            const INT type )
```

1M = 1024∗1024

Allocate, initiate, and check memory

Parameters

size	Number of memory blocks
type	Size of memory blocks

Returns

Void pointer to the allocated memory

Author

Chensong Zhang

Date

2010/08/12

Modified by Chensong Zhang on 07/30/2013: print error if failed

Definition at line 66 of file AuxMemory.c.

9.6.2.2 fasp_mem_free()

```
void fasp_mem_free (
            void * mem )
```

Free up previous allocated memory body.

Parameters

mem	Pointer to the memory body need to be freed

Author

Chensong Zhang

Date

2010/12/24

Definition at line 153 of file AuxMemory.c.

9.6.2.3 fasp_mem_iludata_check()

```
SHORT fasp_mem_iludata_check (
            const ILU_data * iludata )
```

Check wether a ILU_data has enough work space.

Parameters

iludata	Pointer to be cheked

Returns

FASP_SUCCESS if success, else ERROR (negative value)

Author

Xiaozhe Hu, Chensong Zhang

Date

11/27/09

Definition at line 198 of file AuxMemory.c.

9.6.2.4 fasp_mem_realloc()

```
void * fasp_mem_realloc (
            void * oldmem,
            const LONGLONG tsize )
```

Reallocate, initiate, and check memory.

Parameters

oldmem	Pointer to the existing mem block
tsize	Size of memory blocks

Returns

Void pointer to the reallocated memory

Author

Chensong Zhang

Date

2010/08/12

Modified by Chensong Zhang on 07/30/2013: print error if failed

Definition at line 115 of file AuxMemory.c.

9.6.2.5 fasp_mem_usage()

```
void fasp_mem_usage ( )
```

Show total allocated memory currently.

Author

Chensong Zhang

Date

2010/08/12

Definition at line 178 of file AuxMemory.c.

9.6.3 Variable Documentation

9.6.3.1 total_alloc_count

```
unsigned INT total_alloc_count = 0
```

Total allocated memory amount.

total allocation times

Definition at line 44 of file AuxMemory.c.

9.6.3.2 total_alloc_mem

```
unsigned INT total_alloc_mem = 0
```

total allocated memory

Definition at line 43 of file AuxMemory.c.

9.7 AuxMessage.c File Reference

Output some useful messages.

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

Functions

- void print_itinfo (const INT ptrlvl, const INT stop_type, const INT iter, const REAL relres, const REAL absres, const REAL factor)

  Print out iteration information for iterative solvers.
- void print_amgcomplexity (const AMG_data ∗mgl, const SHORT prtlvl)

  Print complexities of AMG method.
- void print_amgcomplexity_bsr (const AMG_data_bsr ∗mgl, const SHORT prtlvl)

  Print complexities of AMG method for BSR matrices.
- void print_cputime (const char ∗message, const REAL cputime)

  Print CPU walltime.
- void print_message (const INT ptrlvl, const char ∗message)

  Print output information if necessary.
- void fasp_chkerr (const SHORT status, const char ∗fctname)

  Check error status and print out error messages before quit.

9.7.1 Detailed Description

Output some useful messages.

Note

This file contains Level-0 (Aux) functions.

9.7.2 Function Documentation

9.7.2.1 fasp_chkerr()

```
void fasp_chkerr (
            const SHORT status,
            const char * fctname )
```

Check error status and print out error messages before quit.

Parameters

status	Error status
fctname	Function name where this routine is called

Author

Chensong Zhang

Date

01/10/2012

Definition at line 205 of file AuxMessage.c.

9.7.2.2 print_amgcomplexity()

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

Print complexities of AMG method.

Parameters

mgl	Multilevel hierachy for AMG
prtlvl	How much information to print

Author

Chensong Zhang

Date

11/16/2009

Definition at line 84 of file AuxMessage.c.

9.7.2.3 print_amgcomplexity_bsr()

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

Print complexities of AMG method for BSR matrices.

Parameters

mgl	Multilevel hierachy for AMG
prtlvl	How much information to print

Author

Chensong Zhang

Date

05/10/2013

Definition at line 128 of file AuxMessage.c.

9.7.2.4 print_cputime()

```
void void print_cputime (
            const char * message,
            const REAL cputime )
```

Print CPU walltime.

Parameters

message	Some string to print out
cputime	Walltime since start to end

Author

Chensong Zhang

Date

04/10/2012

Definition at line 171 of file AuxMessage.c.

9.7.2.5 print_itinfo()

```
void print_itinfo (
            const INT ptrlvl,
            const INT stop_type,
            const INT iter,
            const REAL relres,
            const REAL absres,
            const REAL factor )
```

Print out iteration information for iterative solvers.

Parameters

ptrlvl	Level for output
stop_type	Type of stopping criteria
iter	Number of iterations
relres	Relative residual of different kinds
absres	Absolute residual of different kinds
factor	Contraction factor

Author

Chensong Zhang

Date

11/16/2009

Modified by Chensong Zhang on 03/28/2013: Output initial guess Modified by Chensong Zhang on 04/05/2013: Fix a typo

Definition at line 41 of file AuxMessage.c.

9.7.2.6 print_message()

```
void print_message (
            const INT ptrlvl,
            const char * message )
```

Print output information if necessary.

Parameters

ptrlvl	Level for output
message	Error message to print

Author

Chensong Zhang

Date

11/16/2009

Definition at line 188 of file AuxMessage.c.

9.8 AuxParam.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 (const int argc, const char ∗argv[ ], input_param ∗iniparam)

    Read input from command-line arguments.
- void fasp_param_init (const input_param ∗iniparam, ITS_param ∗itsparam, AMG_param ∗amgparam, ILU_param ∗iluparam, SWZ_param ∗swzparam)

    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 (ITS_param ∗itsparam)

    Initialize ITS_param.
- void fasp_param_ilu_init (ILU_param ∗iluparam)

    Initialize ILU parameters.
- void fasp_param_swz_init (SWZ_param ∗swzparam)

    Initialize Schwarz parameters.
- void fasp_param_amg_set (AMG_param ∗param, const input_param ∗iniparam)

    Set AMG_param from INPUT.
- void fasp_param_ilu_set (ILU_param ∗iluparam, const input_param ∗iniparam)

    Set ILU_param with INPUT.
- void fasp_param_swz_set (SWZ_param ∗swzparam, const input_param ∗iniparam)

    Set SWZ_param with INPUT.
- void fasp_param_solver_set (ITS_param ∗itsparam, const input_param ∗iniparam)

    Set ITS_param with INPUT.
- void fasp_param_amg_to_prec (precond_data ∗pcdata, const AMG_param ∗amgparam)

    Set precond_data with AMG_param.
- void fasp_param_prec_to_amg (AMG_param ∗amgparam, const precond_data ∗pcdata)

    Set AMG_param with precond_data.
- void fasp_param_amg_to_precbsr (precond_data_bsr ∗pcdata, const AMG_param ∗amgparam)

    Set precond_data_bsr with AMG_param.
- void fasp_param_precbsr_to_amg (AMG_param ∗amgparam, const precond_data_bsr ∗pcdata)

    Set AMG_param with precond_data.
- void fasp_param_amg_print (const AMG_param ∗param)

    Print out AMG parameters.
- void fasp_param_ilu_print (const ILU_param ∗param)

    Print out ILU parameters.
- void fasp_param_swz_print (const SWZ_param ∗param)

    Print out Schwarz parameters.
- void fasp_param_solver_print (const ITS_param ∗param)

    Print out itsolver parameters.

9.8.1 Detailed Description

Initialize, set, or print input data and parameters.

Note

This file contains Level-0 (Aux) functions. It requires: AuxInput.c and AuxMessage.c

9.8.2 Function Documentation

9.8.2.1 fasp_param_amg_init()

```
void fasp_param_amg_init (
            AMG_param * amgparam )
```

Initialize AMG parameters.

Parameters

amgparam	Parameters for AMG

Author

Chensong Zhang

Date

2010/04/03

Definition at line 394 of file AuxParam.c.

9.8.2.2 fasp_param_amg_print()

```
void fasp_param_amg_print (
            const AMG_param * param )
```

Print out AMG parameters.

Parameters

param	Parameters for AMG

Author

Chensong Zhang

Date

2010/03/22

Definition at line 804 of file AuxParam.c.

9.8.2.3 fasp_param_amg_set()

```
void fasp_param_amg_set (
            AMG_param * param,
            const input_param * iniparam )
```

Set AMG_param from INPUT.

Parameters

param	Parameters for AMG
iniparam	Input parameters

Author

Chensong Zhang

Date

2010/03/23

Definition at line 522 of file AuxParam.c.

9.8.2.4 fasp_param_amg_to_prec()

```
void fasp_param_amg_to_prec (
            precond_data * pcdata,
            const AMG_param * amgparam )
```

Set precond_data with AMG_param.

Parameters

pcdata	Preconditioning data structure
amgparam	Parameters for AMG

Author

Chensong Zhang

Date

2011/01/10

Definition at line 671 of file AuxParam.c.

9.8.2.5 fasp_param_amg_to_precbsr()

```
void fasp_param_amg_to_precbsr (
            precond_data_bsr * pcdata,
            const AMG_param * amgparam )
```

Set precond_data_bsr with AMG_param.

Parameters

pcdata	Preconditioning data structure
amgparam	Parameters for AMG

Author

Xiaozhe Hu

Date

02/06/2012

Definition at line 739 of file AuxParam.c.

9.8.2.6 fasp_param_ilu_init()

```
void fasp_param_ilu_init (
            ILU_param * iluparam )
```

Initialize ILU parameters.

Parameters

iluparam	Parameters for ILU

Author

Chensong Zhang

Date

2010/04/06

Definition at line 480 of file AuxParam.c.

9.8.2.7 fasp_param_ilu_print()

```
void fasp_param_ilu_print (
            const ILU_param * param )
```

Print out ILU parameters.

Parameters

param	Parameters for ILU

Author

Chensong Zhang

Date

2011/12/20

Definition at line 905 of file AuxParam.c.

9.8.2.8 fasp_param_ilu_set()

```
void fasp_param_ilu_set (
            ILU_param * iluparam,
            const input_param * iniparam )
```

Set ILU_param with INPUT.

Parameters

iluparam	Parameters for ILU
iniparam	Input parameters

Author

Chensong Zhang

Date

2010/04/03

Definition at line 597 of file AuxParam.c.

9.8.2.9 fasp_param_init()

```
void fasp_param_init (
            const input_param * iniparam,
            ITS_param * itsparam,
            AMG_param * amgparam,
            ILU_param * iluparam,
            SWZ_param * swzparam )
```

Initialize parameters, global variables, etc.

Parameters

iniparam	Input parameters
itsparam	Iterative solver parameters
amgparam	AMG parameters
iluparam	ILU parameters
swzparam	Schwarz parameters

Author

Chensong Zhang

Date

2010/08/12

Modified by Chensong Zhang (12/29/2013): rewritten

Definition at line 277 of file AuxParam.c.

9.8.2.10 fasp_param_input_init()

```
void fasp_param_input_init (
            input_param * iniparam )
```

Initialize input parameters.

Parameters

iniparam	Input parameters

Author

Chensong Zhang

Date

2010/03/20

Definition at line 314 of file AuxParam.c.

9.8.2.11 fasp_param_prec_to_amg()

```
void fasp_param_prec_to_amg (
            AMG_param * amgparam,
            const precond_data * pcdata )
```

Set AMG_param with precond_data.

Parameters

amgparam	Parameters for AMG
pcdata	Preconditioning data structure

Author

Chensong Zhang

Date

2011/01/10

Definition at line 706 of file AuxParam.c.

9.8.2.12 fasp_param_precbsr_to_amg()

```
void fasp_param_precbsr_to_amg (
            AMG_param * amgparam,
            const precond_data_bsr * pcdata )
```

Set AMG_param with precond_data.

Parameters

amgparam	Parameters for AMG
pcdata	Preconditioning data structure

Author

Xiaozhe Hu

Date

02/06/2012

Definition at line 774 of file AuxParam.c.

9.8.2.13 fasp_param_set()

```
void fasp_param_set (
            const int argc,
            const char * argv[],
            input_param * iniparam )
```

Read input from command-line arguments.

Parameters

argc	Number of arg input
argv	Input arguments
iniparam	Parameters to be set

Author

Chensong Zhang

Date

12/29/2013

Definition at line 36 of file AuxParam.c.

9.8.2.14 fasp_param_solver_init()

```
void fasp_param_solver_init (
            ITS_param * itsparam )
```

Initialize ITS_param.

Parameters

itsparam	Parameters for iterative solvers

Author

Chensong Zhang

Date

2010/03/23

Definition at line 459 of file AuxParam.c.

9.8.2.15 fasp_param_solver_print()

```
void fasp_param_solver_print (
            const ITS_param * param )
```

Print out itsolver parameters.

Parameters

param	Paramters for iterative solvers

Author

Chensong Zhang

Date

2011/12/20

Definition at line 964 of file AuxParam.c.

9.8.2.16 fasp_param_solver_set()

```
void fasp_param_solver_set (
            ITS_param * itsparam,
            const input_param * iniparam )
```

Set ITS_param with INPUT.

Parameters

itsparam	Parameters for iterative solvers
iniparam	Input parameters

Author

Chensong Zhang

Date

2010/03/23

Definition at line 641 of file AuxParam.c.

9.8.2.17 fasp_param_swz_init()

```
void fasp_param_swz_init (
            SWZ_param * swzparam )
```

Initialize Schwarz parameters.

Parameters

swzparam	Parameters for Schwarz method

Author

Xiaozhe Hu

Date

05/22/2012

Modified by Chensong Zhang on 10/10/2014: Add block solver type

Definition at line 502 of file AuxParam.c.

9.8.2.18 fasp_param_swz_print()

```
void fasp_param_swz_print (
            const SWZ_param * param )
```

Print out Schwarz parameters.

Parameters

param	Parameters for Schwarz

Author

Xiaozhe Hu

Date

05/22/2012

Definition at line 935 of file AuxParam.c.

9.8.2.19 fasp_param_swz_set()

```
void fasp_param_swz_set (
            SWZ_param * swzparam,
            const input_param * iniparam )
```

Set SWZ_param with INPUT.

Parameters

swzparam	Parameters for Schwarz method
iniparam	Input parameters

Author

Xiaozhe Hu

Date

05/22/2012

Definition at line 619 of file AuxParam.c.

9.9 AuxSort.c File Reference

Array sorting/merging and removing duplicated integers.

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

Functions

- INT fasp_aux_BiSearch (const INT nlist, const INT ∗list, const INT value)

  Binary Search.
- INT fasp_aux_unique (INT numbers[ ], const INT size)

  Remove duplicates in an sorted (ascending order) array.
- void fasp_aux_merge (INT numbers[ ], INT work[ ], INT left, INT mid, INT right)

  Merge two sorted arrays.
- void fasp_aux_msort (INT numbers[ ], INT work[ ], INT left, INT right)

  Sort the INT array in ascending order with the merge sort algorithm.
- void fasp_aux_iQuickSort (INT ∗a, INT left, INT right)

  Sort the array (INT type) in ascending order with the quick sorting algorithm.
- void fasp_aux_dQuickSort (REAL ∗a, INT left, INT right)

  Sort the array (REAL type) in ascending order with the quick sorting algorithm.
- void fasp_aux_iQuickSortIndex (INT ∗a, INT left, INT right, INT ∗index)

  Reorder the index of (INT type) so that 'a' is in ascending order.
- void fasp_aux_dQuickSortIndex (REAL ∗a, INT left, INT right, INT ∗index)

  Reorder the index of (REAL type) so that 'a' is ascending in such order.

9.9.1 Detailed Description

Array sorting/merging and removing duplicated integers.

Note

This file contains Level-0 (Aux) functions. It requires: AuxMemory.c

9.9.2 Function Documentation

9.9.2.1 fasp_aux_BiSearch()

```
INT fasp_aux_BiSearch (
            const INT nlist,
            const INT * list,
            const INT value )
```

Binary Search.

Parameters

nlist	Length of the array list
list	Pointer to a set of values
value	The target

Returns

The location of value in array list if succeeded; otherwise, return -1.

Author

Chunsheng Feng

Date

03/01/2011

Definition at line 42 of file AuxSort.c.

9.9.2.2 fasp_aux_dQuickSort()

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

Sort the array (REAL type) in ascending order with the quick sorting algorithm.

Parameters

a	Pointer to the array needed to be sorted
left	Starting index
right	Ending index

Author

Zhiyang Zhou

Date

2009/11/28

Note

'left' and 'right' are usually set to be 0 and n-1, respectively where n is the length of 'a'.

Definition at line 251 of file AuxSort.c.

9.9.2.3 fasp_aux_dQuickSortIndex()

```
void fasp_aux_dQuickSortIndex (
            REAL * a,
            INT left,
            INT right,
            INT * index )
```

Reorder the index of (REAL type) so that 'a' is ascending in such order.

Parameters

a	Pointer to the array
left	Starting index
right	Ending index
index	Index of 'a' (out)

Author

Zhiyang Zhou

Date

2009/12/02

Note

'left' and 'right' are usually set to be 0 and n-1, respectively, where n is the length of 'a'. 'index' should be initialized in the nature order and it has the same length as 'a'.

Definition at line 332 of file AuxSort.c.

9.9.2.4 fasp_aux_iQuickSort()

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

Sort the array (INT type) in ascending order with the quick sorting algorithm.

Parameters

a	Pointer to the array needed to be sorted
left	Starting index
right	Ending index

Author

Zhiyang Zhou

Date

11/28/2009

Note

'left' and 'right' are usually set to be 0 and n-1, respectively where n is the length of 'a'.

Definition at line 213 of file AuxSort.c.

9.9.2.5 fasp_aux_iQuickSortIndex()

```
void fasp_aux_iQuickSortIndex (
            INT * a,
            INT left,
            INT right,
            INT * index )
```

Reorder the index of (INT type) so that 'a' is in ascending order.

Parameters

a	Pointer to the array
left	Starting index
right	Ending index
index	Index of 'a' (out)

Author

Zhiyang Zhou

Date

2009/12/02

Note

'left' and 'right' are usually set to be 0 and n-1,respectively,where n is the length of 'a'. 'index' should be initialized in the nature order and it has the same length as 'a'.

Definition at line 291 of file AuxSort.c.

9.9.2.6 fasp_aux_merge()

```
void fasp_aux_merge (
            INT numbers[ ],
            INT work[ ],
            INT left,
            INT mid,
            INT right )
```

Merge two sorted arrays.

Parameters

numbers	Pointer to the array needed to be sorted
work	Pointer to the work array with same size as numbers
left	Starting index of array 1
mid	Starting index of array 2
right	Ending index of array 1 and 2

Author

Chensong Zhang

Date

11/21/2010

Note

Both arrays are stored in numbers! Arrays should be pre-sorted!

Definition at line 120 of file AuxSort.c.

9.9.2.7 fasp_aux_msort()

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

Sort the INT array in ascending order with the merge sort algorithm.

Parameters

numbers	Pointer to the array needed to be sorted
work	Pointer to the work array with same size as numbers
left	Starting index
right	Ending index

Author

Chensong Zhang

Date

11/21/2010

Note

'left' and 'right' are usually set to be 0 and n-1, respectively

Definition at line 182 of file AuxSort.c.

9.9.2.8 fasp_aux_unique()

```
INT fasp_aux_unique (
            INT numbers[ ],
            const INT size )
```

Remove duplicates in an sorted (ascending order) array.

Parameters

numbers	Pointer to the array needed to be sorted (in/out)
size	Length of the target array

Returns

New size after removing duplicates

Author

Chensong Zhang

Date

11/21/2010

Note

Operation is in place. Does not use any extra or temporary storage.

Definition at line 87 of file AuxSort.c.

9.10 AuxThreads.c File Reference

Get and set number of threads and assign work load for each thread.

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

Functions

- void fasp_get_start_end (const INT procid, const INT nprocs, const INT n, INT *start, INT *end)

  Assign Load to each thread.
- void fasp_set_gs_threads (const INT mythreads, const INT its)

  Set threads for CPR. Please add it at the begin of Krylov OpenMP method function and after iter++.

Variables

- INT THDs_AMG_GS =0
- INT THDs_CPR_lGS =0
- INT THDs_CPR_gGS =0

9.10.1 Detailed Description

Get and set number of threads and assign work load for each thread.

Note

This file contains Level-0 (Aux) functions.

9.10.2 Function Documentation

9.10.2.1 fasp_get_start_end()

```
void fasp_get_start_end (
            const INT procid,
            const INT nprocs,
            const INT n,
            INT * start,
            INT * end )
```

Assign Load to each thread.

Parameters

procid	Index of thread
nprocs	Number of threads
n	Total workload
start	Pointer to the begin of each thread in total workload
end	Pointer to the end of each thread in total workload

Author

Chunsheng Feng, Xiaoqiang Yue and Zheng Li

Date

June/25/2012

Definition at line 91 of file AuxThreads.c.

9.10.2.2 fasp_set_gs_threads()

```
void fasp_set_gs_threads (
            const INT mythreads,
            const INT its )
```

Set threads for CPR. Please add it at the begin of Krylov OpenMP method function and after iter++.

Parameters

mythreads	Total threads of solver
its	Current iteration number in the Krylov methods

Author

Feng Chunsheng, Yue Xiaoqiang

Date

03/20/2011

Definition at line 131 of file AuxThreads.c.

9.10.3 Variable Documentation

9.10.3.1 THDs_AMG_GS

```
INT THDs_AMG_GS =0
```

AMG GS smoothing threads

Definition at line 115 of file AuxThreads.c.

9.10.3.2 THDs_CPR_gGS

```
INT THDs_CPR_gGS =0
```

global matrix GS smoothing threads

Definition at line 117 of file AuxThreads.c.

9.10.3.3 THDs_CPR_lGS

```
INT THDs_CPR_lGS =0
```

reservoir GS smoothing threads

Definition at line 116 of file AuxThreads.c.

9.11 AuxTiming.c File Reference

Timing subroutines.

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

Functions

- void fasp_gettime (REAL ∗time)

    Get system time.

9.11.1 Detailed Description

Timing subroutines.

Note

This file contains Level-0 (Aux) functions.

9.11.2 Function Documentation

9.11.2.1 fasp_gettime()

```
fasp_gettime (
            REAL * time )
```

Get system time.

Author

Chunsheng Feng, Zheng LI

Date

11/10/2012

Modified by Chensong Zhang on 09/22/2014: Use CLOCKS_PER_SEC for cross-platform

Definition at line 35 of file AuxTiming.c.

9.12 AuxVector.c File Reference

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

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

Functions

- SHORT fasp_dvec_isnan (const 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_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 (const dvector ∗x, dvector ∗y)
  Copy dvector x to dvector y.
- REAL fasp_dvec_maxdiff (const dvector ∗x, const dvector ∗y)
  Maximal difference of two dvector x and y.
- void fasp_dvec_symdiagscale (dvector ∗b, const dvector ∗diag)
  Symmetric diagonal scaling D^{-1/2}b.

9.12.1 Detailed Description

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

Note

This file contains Level-0 (Aux) functions. It requires: AuxThreads.c

9.12.2 Function Documentation

9.12.2.1 fasp_dvec_alloc()

```
void fasp_dvec_alloc (
            const INT m,
            dvector * u )
```

Create dvector data space of REAL type.

Parameters

m	Number of rows
u	Pointer to dvector (OUTPUT)

Author

Chensong Zhang

Date

2010/04/06

Definition at line 105 of file AuxVector.c.

9.12.2.2 fasp_dvec_cp()

```
void fasp_dvec_cp (
            const dvector * x,
            dvector * y )
```

Copy dvector x to dvector y.

Parameters

x	Pointer to dvector
y	Pointer to dvector (MODIFIED)

Author

Chensong Zhang

Date

11/16/2009

Definition at line 334 of file AuxVector.c.

9.12.2.3 fasp_dvec_create()

```
dvector fasp_dvec_create (
            const INT m )
```

Create dvector data space of REAL type.

Parameters

m	Number of rows

Returns

u The new dvector

Author

Chensong Zhang

Date

2010/04/06

Definition at line 62 of file AuxVector.c.

9.12.2.4 fasp_dvec_free()

```
void fasp_dvec_free (
            dvector * u )
```

Free vector data space of REAL type.

Parameters

u	Pointer to dvector which needs to be deallocated

Author

Chensong Zhang

Date

2010/04/03

Definition at line 145 of file AuxVector.c.

9.12.2.5 fasp_dvec_isnan()

```
SHORT fasp_dvec_isnan (
            const 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 39 of file AuxVector.c.

9.12.2.6 fasp_dvec_maxdiff()

```
REAL fasp_dvec_maxdiff (
            const dvector * x,
            const 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 357 of file AuxVector.c.

9.12.2.7 fasp_dvec_rand()

```
void fasp_dvec_rand (
            const INT n,
            dvector * x )
```

Generate random REAL vector in the range from 0 to 1.

Parameters

n	Size of the vector
x	Pointer to dvector

Note

Sample usage:

dvector xapp;

fasp_dvec_create(100,&xapp);

fasp_dvec_rand(100,&xapp);

fasp_dvec_print(100,&xapp);

Author

Chensong Zhang

Date

11/16/2009

Definition at line 194 of file AuxVector.c.

9.12.2.8 fasp_dvec_set()

```
void fasp_dvec_set (
            INT n,
            dvector * x,
            REAL val )
```

Initialize dvector x[i]=val for i=0:n-1.

Parameters

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

Author

Chensong Zhang

Date

11/16/2009

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

Definition at line 224 of file AuxVector.c.

9.12.2.9 fasp_dvec_symdiagscale()

```
void fasp_dvec_symdiagscale (
            dvector * b,
            const dvector * diag )
```

Symmetric diagonal scaling D^{-1/2}b.

Parameters

b	Pointer to dvector
diag	Pointer to dvector: the diagonal entries

Author

Xiaozhe Hu

Date

01/31/2011

Definition at line 410 of file AuxVector.c.

9.12.2.10 fasp_ivec_alloc()

```
void fasp_ivec_alloc (
            const INT m,
            ivector * u )
```

Create vector data space of INT type.

Parameters

m	Number of rows
u	Pointer to ivector (OUTPUT)

Author

Chensong Zhang

Date

2010/04/06

Definition at line 125 of file AuxVector.c.

9.12.2.11 fasp_ivec_create()

```
ivector fasp_ivec_create (
            const INT m )
```

Create vector data space of INT type.

Parameters

m	Number of rows

Returns

u The new ivector

Author

Chensong Zhang

Date

2010/04/06

Definition at line 84 of file AuxVector.c.

9.12.2.12 fasp_ivec_free()

```
void fasp_ivec_free (
            ivector * u )
```

Free vector data space of INT type.

Parameters

u	Pointer to ivector which needs to be deallocated

Author

Chensong Zhang

Date

2010/04/03

Note

This function is same as fasp_dvec_free except input type.

Definition at line 165 of file AuxVector.c.

9.12.2.13 fasp_ivec_set()

```
void fasp_ivec_set (
            const INT m,
            ivector * u )
```

Set ivector value to be m.

Parameters

m	Integer value of ivector
u	Pointer to ivector (MODIFIED)

Author

Chensong Zhang

Date

04/03/2010

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

Definition at line 292 of file AuxVector.c.

9.13 BlaArray.c File Reference

BLAS1 operations for arrays.

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

Functions

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

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

  y = a∗x + y
- void fasp_blas_darray_axpy_nc2 (const REAL a, const REAL ∗x, REAL ∗y)

  y = a∗x + y, length of x and y should be 2
- void fasp_blas_darray_axpy_nc3 (const REAL a, const REAL ∗x, REAL ∗y)

  y = a∗x + y, length of x and y should be 3
- void fasp_blas_darray_axpy_nc5 (const REAL a, const REAL ∗x, REAL ∗y)

  y = a∗x + y, length of x and y should be 5
- void fasp_blas_darray_axpy_nc7 (const REAL a, const REAL ∗x, REAL ∗y)

  y = a∗x + y, length of x and y should be 7
- void fasp_blas_darray_axpyz (const INT n, const REAL a, const REAL ∗x, const REAL ∗y, REAL ∗z)

  z = a∗x + y
- void fasp_blas_darray_axpyz_nc2 (const REAL a, const REAL ∗x, const REAL ∗y, REAL ∗z)

  z = a∗x + y, length of x, y and z should be 2
- void fasp_blas_darray_axpyz_nc3 (const REAL a, const REAL ∗x, const REAL ∗y, REAL ∗z)

z = a∗x + y, length of x, y and z should be 3
- void fasp_blas_darray_axpyz_nc5 (const REAL a, const REAL ∗x, const REAL ∗y, REAL ∗z)

    z = a∗x + y, length of x, y and z should be 5
- void fasp_blas_darray_axpyz_nc7 (const REAL a, const REAL ∗x, const REAL ∗y, REAL ∗z)

    z = a∗x + y, length of x, y and z should be 7
- void fasp_blas_darray_axpby (const INT n, const REAL a, const REAL ∗x, const REAL b, REAL ∗y)

    y = a∗x + b∗y
- REAL fasp_blas_darray_norm1 (const INT n, const REAL ∗x)

    L1 norm of array x.
- REAL fasp_blas_darray_norm2 (const INT n, const REAL ∗x)

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

    Linf norm of array x.
- REAL fasp_blas_darray_dotprod (const INT n, const REAL ∗x, const REAL ∗y)

    Inner product of two arraies x and y.

9.13.1 Detailed Description

BLAS1 operations for arrays.

Note

This file contains Level-1 (Bla) functions. It requires: AuxThreads.c

9.13.2 Function Documentation

9.13.2.1 fasp_blas_darray_ax()

```
void fasp_blas_darray_ax (
            const INT n,
            const REAL a,
            REAL * x )  [inline]
```

x = a∗x

Parameters

n	Number of variables
a	Factor a
x	Pointer to x

Author

Chensong Zhang

Date

07/01/2009

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

Warning

x is reused to store the resulting array.

Definition at line 43 of file BlaArray.c.

9.13.2.2 fasp_blas_darray_axpby()

```
void fasp_blas_darray_axpby (
            const INT n,
            const REAL a,
            const 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, reused to store the resulting array

Author

Chensong Zhang

Date

07/01/2009

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

Definition at line 580 of file BlaArray.c.

9.13.2.3 fasp_blas_darray_axpy()

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

y = a*x + y

Parameters

n	Number of variables
a	Factor a
x	Pointer to x
y	Pointer to y, reused to store the resulting array

Author

Chensong Zhang

Date

07/01/2009

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

Definition at line 93 of file BlaArray.c.

9.13.2.4 fasp_blas_darray_axpy_nc2()

```
void fasp_blas_darray_axpy_nc2 (
            const REAL a,
            const REAL * x,
            REAL * y )
```

y = a*x + y, length of x and y should be 2

Parameters

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

Author

Xiaozhe Hu

Date

18/11/2011

Definition at line 170 of file BlaArray.c.

9.13.2.5 fasp_blas_darray_axpy_nc3()

```
void fasp_blas_darray_axpy_nc3 (
            const REAL a,
            const REAL * x,
            REAL * y )
```

y = a*x + y, length of x and y should be 3

Parameters

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

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 193 of file BlaArray.c.

9.13.2.6 fasp_blas_darray_axpy_nc5()

```
void fasp_blas_darray_axpy_nc5 (
            const REAL a,
            const REAL * x,
            REAL * y )
```

y = a*x + y, length of x and y should be 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 222 of file BlaArray.c.

9.13.2.7 fasp_blas_darray_axpy_nc7()

```
void fasp_blas_darray_axpy_nc7 (
            const REAL a,
            const REAL * x,
            REAL * y )
```

y = a∗x + y, length of x and y should be 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 269 of file BlaArray.c.

9.13.2.8 fasp_blas_darray_axpyz()

```
void fasp_blas_darray_axpyz (
            const INT n,
            const REAL a,
            const REAL * x,
            const 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 347 of file BlaArray.c.

9.13.2.9 fasp_blas_darray_axpyz_nc2()

```
void fasp_blas_darray_axpyz_nc2 (
            const REAL a,
            const REAL * x,
            const REAL * y,
            REAL * z )
```

z = a∗x + y, length of x, y and z should be 2

Parameters

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

Author

Xiaozhe Hu

Date

18/11/2011

Definition at line 393 of file BlaArray.c.

9.13.2.10 fasp_blas_darray_axpyz_nc3()

```
void fasp_blas_darray_axpyz_nc3 (
            const REAL a,
            const REAL * x,
            const REAL * y,
            REAL * z )
```

z = a∗x + y, length of x, y and z should be 3

Parameters

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

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 419 of file BlaArray.c.

9.13.2.11 fasp_blas_darray_axpyz_nc5()

```
void fasp_blas_darray_axpyz_nc5 (
            const REAL a,
            const REAL * x,
            const REAL * y,
            REAL * z )
```

z = a∗x + y, length of x, y and z should be 5

Parameters

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

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 451 of file BlaArray.c.

9.13.2.12 fasp_blas_darray_axpyz_nc7()

```
void fasp_blas_darray_axpyz_nc7 (
            const REAL a,
            const REAL * x,
            const REAL * y,
            REAL * z )
```

z = a∗x + y, length of x, y and z should be 7

Parameters

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

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 501 of file BlaArray.c.

9.13.2.13 fasp_blas_darray_dotprod()

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

Inner product of two arraies x and 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 741 of file BlaArray.c.

9.13.2.14 fasp_blas_darray_norm1()

```
REAL fasp_blas_darray_norm1 (
            const INT n,
            const REAL * x )
```

L1 norm of array x.

Parameters

n	Number of variables
x	Pointer to x

Returns

L1 norm of x

Author

Chensong Zhang

Date

07/01/2009

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

Definition at line 628 of file BlaArray.c.

9.13.2.15 fasp_blas_darray_norm2()

```
REAL fasp_blas_darray_norm2 (
            const INT n,
            const REAL * x )
```

L2 norm of array x.

Parameters

n	Number of variables
x	Pointer to x

Returns

L2 norm of x

Author

Chensong Zhang

Date

07/01/2009

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

Definition at line 657 of file BlaArray.c.

9.13.2.16 fasp_blas_darray_norminf()

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

Linf norm of array x.

Parameters

n	Number of variables
x	Pointer to x

Returns

L_inf norm of x

Author

Chensong Zhang

Date

07/01/2009

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

Definition at line 686 of file BlaArray.c.

9.14 BlaEigen.c File Reference

Computing the extreme eigenvalues.

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

Functions

- REAL fasp_dcsr_maxeig (const dCSRmat ∗A, const REAL tol, const INT maxit)

  Approximate the largest eigenvalue of A by the power method.

9.14.1 Detailed Description

Computing the extreme eigenvalues.

Note

This file contains Level-1 (Bla) functions. It requires: AuxVector.c, BlaArray.c, BlaSpmvCSR.c, and BlaVector.c

9.14.2 Function Documentation

9.14.2.1 fasp_dcsr_maxeig()

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

Approximate the largest eigenvalue of A by the power method.

Parameters

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

Returns

Largest eigenvalue

Author

Xiaozhe Hu

Date

01/25/2011

Definition at line 37 of file BlaEigen.c.

9.15 BlaFormat.c File Reference

Subroutines for matrix format conversion.

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

Functions

- SHORT fasp_format_dcoo_dcsr (const dCOOmat *A, dCSRmat *B)

  Transform a REAL matrix from its IJ format to its CSR format.
- SHORT fasp_format_dcsr_dcoo (const dCSRmat *A, dCOOmat *B)

  Transform a REAL matrix from its CSR format to its IJ format.
- SHORT fasp_format_dstr_dcsr (const dSTRmat *A, dCSRmat *B)

  Transfer a 'dSTRmat' type matrix into a 'dCSRmat' type matrix.
- dCSRmat fasp_format_dblc_dcsr (const dBLCmat *Ab)

  Form the whole dCSRmat A using blocks given in Ab.
- dCSRLmat * fasp_format_dcsrl_dcsr (const dCSRmat *A)

  Convert a dCSRmat into a dCSRLmat.
- dCSRmat fasp_format_dbsr_dcsr (const dBSRmat *B)

  Transfer a 'dBSRmat' type matrix into a dCSRmat.
- dBSRmat fasp_format_dcsr_dbsr (const dCSRmat *A, const INT nb)

  Transfer a dCSRmat type matrix into a dBSRmat.
- dBSRmat fasp_format_dstr_dbsr (const dSTRmat *B)

  Transfer a 'dSTRmat' type matrix to a 'dBSRmat' type matrix.
- dCOOmat * fasp_format_dbsr_dcoo (const dBSRmat *B)

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

9.15.1 Detailed Description

Subroutines for matrix format conversion.

Note

This file contains Level-1 (Bla) functions. It requires: AuxArray.c, AuxMemory.c, AuxThreads.c, BlaSparseBSR.c, BlaSparseCSR.c, and BlaSparseCSRL.c

9.15.2 Function Documentation

9.15.2.1 fasp_format_dblc_dcsr()

```
dCSRmat fasp_format_dblc_dcsr (
        const dBLCmat * Ab )
```

Form the whole dCSRmat A using blocks given in Ab.

Parameters

Ab	Pointer to dBLCmat matrix

Returns

dCSRmat matrix if succeed, NULL if fail

Author

Shiquan Zhang

Date

08/10/2010

Definition at line 294 of file BlaFormat.c.

9.15.2.2 fasp_format_dbsr_dcoo()

```
dCOOmat * fasp_format_dbsr_dcoo (
            const dBSRmat * B )
```

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

Parameters

B	Pointer to dBSRmat matrix

Returns

Pointer to dCOOmat matrix

Author

Zhiyang Zhou

Date

2010/10/26

Definition at line 945 of file BlaFormat.c.

9.15.2.3 fasp_format_dbsr_dcsr()

```
dCSRmat fasp_format_dbsr_dcsr (
            const dBSRmat * B )
```

Transfer a 'dBSRmat' type matrix into a dCSRmat.

Parameters

B	Pointer to dBSRmat matrix

Returns

dCSRmat matrix

Author

Zhiyang Zhou

Date

10/23/2010

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

Note

Works for general nb (Xiaozhe)

Definition at line 497 of file BlaFormat.c.

9.15.2.4 fasp_format_dcoo_dcsr()

```
SHORT fasp_format_dcoo_dcsr (
            const dCOOmat * A,
            dCSRmat * B )
```

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

Parameters

A	Pointer to dCOOmat matrix
B	Pointer to dCSRmat matrix

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Xuehai Huang

Date

08/10/2009

Definition at line 36 of file BlaFormat.c.

9.15.2.5 fasp_format_dcsr_dbsr()

```
dBSRmat fasp_format_dcsr_dbsr (
            const dCSRmat * A,
            const INT nb )
```

Transfer a dCSRmat type matrix into a dBSRmat.

Parameters

A	Pointer to the dCSRmat type matrix
nb	size of each block

Returns

dBSRmat matrix

Author

Zheng Li

Date

03/27/2014

Note

modified by Xiaozhe Hu to avoid potential memory leakage problem

Definition at line 723 of file BlaFormat.c.

9.15.2.6 fasp_format_dcsr_dcoo()

```
SHORT fasp_format_dcsr_dcoo (
            const dCSRmat * A,
            dCOOmat * B )
```

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

Parameters

A	Pointer to dCSRmat matrix
B	Pointer to dCOOmat matrix

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Xuehai Huang

Date

08/10/2009

Modified by Chunsheng Feng, Zheng Li on 10/12/2012

Definition at line 83 of file BlaFormat.c.

9.15.2.7 fasp_format_dcsrl_dcsr()

```
dCSRLmat * fasp_format_dcsrl_dcsr (
            const dCSRmat * A )
```

Convert a dCSRmat into a dCSRLmat.

Parameters

A	Pointer to dCSRLmat matrix

Returns

Pointer to dCSRLmat matrix

Author

Zhiyang Zhou

Date

2011/01/07

Definition at line 363 of file BlaFormat.c.

9.15.2.8 fasp_format_dstr_dbsr()

```
dBSRmat fasp_format_dstr_dbsr (
            const dSTRmat * B )
```

Transfer a 'dSTRmat' type matrix to a 'dBSRmat' type matrix.

Parameters

B	Pointer to dSTRmat matrix

Returns

dBSRmat matrix

Author

Zhiyang Zhou

Date

2010/10/26

Definition at line 841 of file BlaFormat.c.

9.15.2.9 fasp_format_dstr_dcsr()

```
SHORT fasp_format_dstr_dcsr (
            const dSTRmat * A,
            dCSRmat * B )
```

Transfer a 'dSTRmat' type matrix into a 'dCSRmat' type matrix.

Parameters

A	Pointer to dSTRmat matrix
B	Pointer to dCSRmat matrix

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Zhiyang Zhou

Date

2010/04/29

Definition at line 119 of file BlaFormat.c.

9.16 BlaILU.c File Reference

Incomplete LU decomposition: ILUk, ILUt, ILUtp.

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

Functions

- void fasp_iluk (INT n, REAL ∗a, INT ∗ja, INT ∗ia, INT lfil, REAL ∗alu, INT ∗jlu, INT iwk, INT ∗ierr, INT ∗nzlu)

  Get ILU factorization with level of fill-in k (ilu(k)) for a CSR matrix A.
- void fasp_ilut (INT n, REAL ∗a, INT ∗ja, INT ∗ia, INT lfil, REAL droptol, REAL ∗alu, INT ∗jlu, INT iwk, INT ∗ierr, INT ∗nz)

  Get incomplete LU factorization with dual truncations of a CSR matrix A.
- void fasp_ilutp (INT n, REAL ∗a, INT ∗ja, INT ∗ia, INT lfil, REAL droptol, REAL permtol, INT mbloc, REAL ∗alu, INT ∗jlu, INT iwk, INT ∗ierr, INT ∗nz)

  Get incomplete LU factorization with pivoting dual truncations of a CSR matrix A.
- void fasp_symbfactor (INT n, INT ∗colind, INT ∗rwptr, INT levfill, INT nzmax, INT ∗nzlu, INT ∗ijlu, INT ∗uptr, INT ∗ierr)

  Symbolic factorization of a CSR matrix A in compressed sparse row format, with resulting factors stored in a single MSR data structure.

9.16.1 Detailed Description

Incomplete LU decomposition: ILUk, ILUt, ILUtp.

Note

This file contains Level-1 (Bla) functions. It requires: AuxMemory.c

Translated from SparseKit (Fortran code) by Chunsheng Feng, 09/03/2016

9.16.2 Function Documentation

9.16.2.1 fasp_iluk()

```
void fasp_iluk (
            INT n,
            REAL * a,
            INT * ja,
            INT * ia,
            INT lfil,
            REAL * alu,
            INT * jlu,
            INT iwk,
            INT * ierr,
            INT * nzlu )
```

Get ILU factorization with level of fill-in k (ilu(k)) for a CSR matrix A.

Parameters

n	row number of A
a	nonzero entries of A
ja	integer array of column for A
ia	integer array of row pointers for A
lfil	integer. The fill-in parameter. Each row of L and each row of U will have a maximum of lfil elements (excluding the diagonal element). lfil must be .ge. 0.
alu	matrix stored in Modified Sparse Row (MSR) format containing the L and U factors together. The diagonal (stored in alu(1:n)) is inverted. Each i-th row of the alu,jlu matrix contains the i-th row of L (excluding the diagonal entry=1) followed by the i-th row of U.
jlu	integer array of length n containing the pointers to the beginning of each row of U in the matrix alu,jlu.
iwk	integer. The minimum length of arrays alu, jlu, and levs.
ierr	integer pointer. Return error message with the following meaning. 0 –> successful return. >0 –> zero pivot encountered at step number ierr. -1 –> Error. input matrix may be wrong. (The elimination process has generated a row in L or U whose length is .gt. n.) -2 –> The matrix L overflows the array al. -3 –> The matrix U overflows the array alu. -4 –> Illegal value for lfil. -5 –> zero row encountered.
nzlu	integer pointer. Return number of nonzero entries for alu and jlu

Note

: All the diagonal elements of the input matrix must be nonzero.

Author

Chunsheng Feng

Date

09/06/2016

Definition at line 71 of file BlaILU.c.

9.16.2.2 fasp_ilut()

```
void fasp_ilut (
        INT n,
        REAL * a,
        INT * ja,
        INT * ia,
        INT lfil,
        REAL droptol,
        REAL * alu,
        INT * jlu,
        INT iwk,
        INT * ierr,
        INT * nz )
```

Get incomplete LU factorization with dual truncations of a CSR matrix A.

Parameters

n	row number of A
a	nonzero entries of A
ja	integer array of column for A
ia	integer array of row pointers for A
lfil	integer. The fill-in parameter. Each row of L and each row of U will have a maximum of lfil elements (excluding the diagonal element). lfil must be .ge. 0.
droptol	real∗8. Sets the threshold for dropping small terms in the factorization. See below for details on dropping strategy.
alu	matrix stored in Modified Sparse Row (MSR) format containing the L and U factors together. The diagonal (stored in alu(1:n)) is inverted. Each i-th row of the alu,jlu matrix contains the i-th row of L (excluding the diagonal entry=1) followed by the i-th row of U.
jlu	integer array of length n containing the pointers to the beginning of each row of U in the matrix alu,jlu.
iwk	integer. The lengths of arrays alu and jlu. If the arrays are not big enough to store the ILU factorizations, ilut will stop with an error message.
ierr	integer pointer. Return error message with the following meaning. 0 –> successful return. >0 –> zero pivot encountered at step number ierr. -1 –> Error. input matrix may be wrong. (The elimination process has generated a row in L or U whose length is .gt. n.) -2 –> The matrix L overflows the array al. -3 –> The matrix U overflows the array alu. -4 –> Illegal value for lfil. -5 –> zero row encountered.
nz	integer pointer. Return number of nonzero entries for alu and jlu

Note

All the diagonal elements of the input matrix must be nonzero.

Author

Chunsheng Feng

Date

09/06/2016

Definition at line 465 of file BlaILU.c.

9.16.2.3 fasp_ilutp()

```
void fasp_ilutp (
            INT n,
            REAL * a,
            INT * ja,
            INT * ia,
            INT lfil,
            REAL droptol,
            REAL permtol,
            INT mbloc,
            REAL * alu,
            INT * jlu,
            INT iwk,
            INT * ierr,
            INT * nz )
```

Get incomplete LU factorization with pivoting dual truncations of a CSR matrix A.

Parameters

n	row number of A
a	nonzero entries of A
ja	integer array of column for A
ia	integer array of row pointers for A
lfil	integer. The fill-in parameter. Each row of L and each row of U will have a maximum of lfil elements (excluding the diagonal element). lfil must be .ge. 0.
droptol	real*8. Sets the threshold for dropping small terms in the factorization. See below for details on dropping strategy.
permtol	tolerance ratio used to determne whether or not to permute two columns. At step i columns i and j are permuted when abs(a(i,j))*permtol .gt. abs(a(i,i)) [0 –> never permute; good values 0.1 to 0.01]
mbloc	integer.If desired, permuting can be done only within the diagonal blocks of size mbloc. Useful for PDE problems with several degrees of freedom.. If feature not wanted take mbloc=n.

Parameters

alu	matrix stored in Modified Sparse Row (MSR) format containing the L and U factors together. The diagonal (stored in alu(1:n)) is inverted. Each i-th row of the alu,jlu matrix contains the i-th row of L (excluding the diagonal entry=1) followed by the i-th row of U.
jlu	integer array of length n containing the pointers to the beginning of each row of U in the matrix alu,jlu.
iwk	integer. The lengths of arrays alu and jlu. If the arrays are not big enough to store the ILU factorizations, ilut will stop with an error message.
ierr	integer pointer. Return error message with the following meaning. 0 –> successful return. >0 –> zero pivot encountered at step number ierr. -1 –> Error. input matrix may be wrong. (The elimination process has generated a row in L or U whose length is .gt. n.) -2 –> The matrix L overflows the array al. -3 –> The matrix U overflows the array alu. -4 –> Illegal value for lfil. -5 –> zero row encountered.
nz	integer pointer. Return number of nonzero entries for alu and jlu

Note

: All the diagonal elements of the input matrix must be nonzero.

Author

Chunsheng Feng

Date

09/06/2016

Definition at line 902 of file BlaILU.c.

9.16.2.4 fasp_symbfactor()

```
void fasp_symbfactor (
            INT n,
            INT * colind,
            INT * rwptr,
            INT levfill,
            INT nzmax,
            INT * nzlu,
            INT * ijlu,
            INT * uptr,
            INT * ierr )
```

Symbolic factorization of a CSR matrix A in compressed sparse row format, with resulting factors stored in a single MSR data structure.

Parameters

n	row number of A
colind	integer array of column for A
rwptr	integer array of row pointers for A
levfill	integer. Level of fill-in allowed
nzmax	integer. The maximum number of nonzero entries in the approximate factorization of a. This is the amount of storage allocated for ijlu.
nzlu	integer pointer. Return number of nonzero entries for alu and jlu
ijlu	integer array of length nzlu containing pointers to delimit rows and specify column number for stored elements of the approximate factors of A. the L and U factors are stored as one matrix.
uptr	integer array of length n containing the pointers to upper trig matrix
ierr	integer pointer. Return error message with the following meaning. 0 –> successful return. 1 –> not enough storage; check mneed.

Author

Chunsheng Feng

Date

09/06/2016

Symbolic factorization of a matrix in compressed sparse row format, ∗ with resulting factors stored in a single MSR data structure. ∗

This routine uses the CSR data structure of A in two integer vectors ∗ colind, rwptr to set up the data structure for the ILU(levfill) ∗ factorization of A in the integer vectors ijlu and uptr. Both L ∗ and U are stored in the same structure, and uptr(i) is the pointer ∗ to the beginning of the i-th row of U in ijlu. ∗

Method Used ∗ =========== ∗

The implementation assumes that the diagonal entries are ∗ nonzero, and remain nonzero throughout the elimination ∗ process. The algorithm proceeds row by row. When computing ∗ the sparsity pattern of the i-th row, the effect of row ∗ operations from previous rows is considered. Only those ∗ preceding rows j for which (i,j) is nonzero need be considered, ∗ since otherwise we would not have formed a linear combination ∗ of rows i and j. ∗

The method used has some variations possible. The definition ∗ of ILU(s) is not well specified enough to get a factorization ∗ that is uniquely defined, even in the sparsity pattern that ∗ results. For s = 0 or 1, there is not much variation, but for ∗ higher levels of fill the problem is as follows: Suppose ∗ during the decomposition while computing the nonzero pattern ∗ for row i the following principal submatrix is obtained: ∗ _______________________ ∗ | | | ∗ | | | ∗ | j,j | j,k | ∗ | | | ∗ |___________|___________| ∗ | | | ∗ | | | ∗ | i,j | i,k | ∗ | | | ∗ |___________|___________| ∗

Furthermore, suppose that entry (i,j) resulted from an earlier ∗ fill-in and has level s1, and (j,k) resulted from an earlier ∗ fill-in and has level s2: ∗ _______________________ ∗ | | | ∗ | | | ∗ | level 0 | level s2 | ∗ | | | ∗ |___________|______↩ ______| ∗ | | | ∗ | | | ∗ | level s1 | | ∗ | | | ∗ |___________|___________| ∗

When using A(j,j) to annihilate A(i,j), fill-in will be incurred ∗ in A(i,k). How should its level be defined? It would not be ∗ operated on if A(i,j) or A(j,m) had not been filled in. The ∗ version used here is to define its level as s1 + s2 +

1. However, ∗ other reasonable choices would have been min(s1,s2) or max(s1,s2). ∗ Using the sum gives a more conservative strategy in terms of the ∗ growth of the number of nonzeros as s increases. ∗

levels(n+2:nzlu) stores the levels from previous rows, ∗ that is, the s2's above. levels(1:n) stores the fill-levels ∗ of the current row (row i), which are the s1's above. ∗ levels(n+1) is not used, so levels is conformant with MSR format. ∗

Vectors used: ∗ ============= ∗

lastcol(n): ∗ The integer lastcol(k) is the row index of the last row ∗ to have a nonzero in column k, including the current ∗ row, and fill-in up to this point. So for the matrix ∗

|-----------------------—| ∗ | 11 12 15 | ∗ | 21 22 26| ∗ | 32 33 34 | ∗ | 41 43 44 | ∗ | 52 54 55 56| ∗ | 62 66| ∗ ------------------------—- ∗

after step 1, lastcol() = [1 0 0 0 1 0] ∗ after step 2, lastcol() = [2 2 0 0 2 2] ∗ after step 3, lastcol() = [2 3 3 3 2 3] ∗ after step 4, lastcol() = [4 3 4 4 4 3] ∗ after step 5, lastcol() = [4 5 4 5 5 5] ∗ after step 6, lastcol() = [4 6 4 5 5 6] ∗

Note that on step 2, lastcol(5) = 2 because there is a ∗ fillin position (2,5) in the matrix. lastcol() is used ∗ to determine if a nonzero occurs in column j because ∗ it is a nonzero in the original matrix, or was a fill. ∗

rowll(n): ∗ The integer vector rowll is used to keep a linked list of ∗ the nonzeros in the current row, allowing fill-in to be ∗ introduced sensibly. rowll is initialized with the ∗ original nonzeros of the current row, and then sorted ∗ using a shell sort. A pointer called head ∗ (what ingenuity) is initialized. Note that at any ∗ point rowll may contain garbage left over from previous ∗ rows, which the linked list structure skips over. ∗ For row 4 of the matrix above, first rowll is set to ∗ rowll() = [3 1 2 5 - -], where - indicates any integer. ∗ Then the vector is sorted, which yields ∗ rowll() = [1 2 3 5 - -]. The vector is then expanded ∗ to linked list form by setting head = 1 and ∗ rowll() = [2 3 5 - 7 -], where 7 indicates termination. ∗

ijlu(nzlu): ∗ The returned nonzero structure for the LU factors. ∗ This is built up row by row in MSR format, with both L ∗ and U stored in the data structure. Another vector, uptr(n), ∗ is used to give pointers to the beginning of the upper ∗ triangular part of the LU factors in ijlu. ∗

levels(n+2:nzlu): ∗ This vector stores the fill level for each entry from ∗ all the previous rows, used to compute if the current entry ∗ will exceed the allowed levels of fill. The value in ∗ levels(m) is added to the level of fill for the element in ∗ the current row that is being reduced, to figure if ∗ a column entry is to be accepted as fill, or rejected. ∗ See the method explanation above. ∗

levels(1:n): ∗ This vector stores the fill level number for the current ∗ row's entries. If they were created as fill elements ∗ themselves, this number is added to the corresponding ∗ entry in levels(n+2:nzlu) to see if a particular column ∗ entry will ∗ be created as new fill or not. NOTE: in practice, the ∗ value in levels(1:n) is one larger than the "fill" level of ∗ the corresponding row entry, except for the diagonal ∗ entry. That is why the accept/reject test in the code ∗ is "if (levels(j) + levels(m) .le. levfill + 1)". ∗

on entry:

n = The order of the matrix A. ija = Integer array. Matrix A stored in modified sparse row format. levfill = Integer. Level of fill-in allowed. nzmax = Integer. The maximum number of nonzero entries in the approximate factorization of a. This is the amount of storage allocated for ijlu.

on return:

nzlu = The actual number of entries in the approximate factors, plus one. ijlu = Integer array of length nzlu containing pointers to delimit rows and specify column number for stored elements of the approximate factors of a. the l and u factors are stored as one matrix. uptr = Integer array of length n containing the pointers to upper trig matrix

ierr is an error flag: ierr = -i –> near zero pivot in step i ierr = 0 –> all's OK ierr = 1 –> not enough storage; check mneed. ierr = 2 –> illegal parameter

mneed = contains the actual number of elements in ldu, or the amount of additional storage needed for ldu

work arrays:

lastcol = integer array of length n containing last update of the corresponding column. levels = integer array of length n containing the level of fill-in in current row in its first n entries, and level of fill of previous rows of U in remaining part. rowll = integer array of length n containing pointers to implement a linked list for the fill-in elements.

external functions:

ifix, float, min0, srtr

Definition at line 1368 of file BlaILU.c.

9.17 BlaILUSetupBSR.c File Reference

Setup incomplete LU decomposition for dBSRmat matrices.

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

Functions

- SHORT fasp_ilu_dbsr_setup (dBSRmat *A, ILU_data *iludata, ILU_param *iluparam)

  Get ILU decoposition of a BSR matrix A.
- SHORT fasp_ilu_dbsr_setup_omp (dBSRmat *A, ILU_data *iludata, ILU_param *iluparam)

  Multi-threads parallel ILU decoposition of a BSR matrix A based on graph coloring.
- SHORT fasp_ilu_dbsr_setup_levsch_omp (dBSRmat *A, ILU_data *iludata, ILU_param *iluparam)

  Get ILU decoposition of a BSR matrix A based on level schedule strategy.
- SHORT fasp_ilu_dbsr_setup_mc_omp (dBSRmat *A, dCSRmat *Ap, ILU_data *iludata, ILU_param *iluparam)

  Multi-threads parallel ILU decoposition of a BSR matrix A based on graph coloring.

9.17.1 Detailed Description

Setup incomplete LU decomposition for dBSRmat matrices.

Note

This file contains Level-1 (Bla) functions. It requires: AuxArray.c, AuxMemory.c, AuxTiming.c, BlaSmallMatInv.c, BlaILU.c, BlaSmallMat.c, BlaSmallMatInv.c, BlaSparseBSR.c, BlaSparseCSR.c, BlaSpmvCSR.c, and PreData↩Init.c

9.17.2 Function Documentation

9.17.2.1 fasp_ilu_dbsr_setup()

```
SHORT fasp_ilu_dbsr_setup (
            dBSRmat * A,
            ILU_data * iludata,
            ILU_param * iluparam )
```

Get ILU decoposition of a BSR matrix A.

Parameters

A	Pointer to dBSRmat matrix
iludata	Pointer to ILU_data
iluparam	Pointer to ILU_param

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Shiquan Zhang, Xiaozhe Hu

Date

11/08/2010

Note

Works for general nb (Xiaozhe)
Change the size of work space by Zheng Li 04/26/2015.

Definition at line 53 of file BlaILUSetupBSR.c.

9.17.2.2 fasp_ilu_dbsr_setup_levsch_omp()

```
SHORT fasp_ilu_dbsr_setup_levsch_omp (
            dBSRmat * A,
            ILU_data * iludata,
            ILU_param * iluparam )
```

Get ILU decoposition of a BSR matrix A based on level schedule strategy.

Parameters

A	Pointer to dBSRmat matrix
iludata	Pointer to ILU_data
iluparam	Pointer to ILU_param

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Zheng Li

Date

12/04/2016

Note

Only works for 1, 2, 3 nb (Zheng)

Definition at line 276 of file BlaILUSetupBSR.c.

9.17.2.3 fasp_ilu_dbsr_setup_mc_omp()

```
SHORT fasp_ilu_dbsr_setup_mc_omp (
            dBSRmat * A,
            dCSRmat * Ap,
            ILU_data * iludata,
            ILU_param * iluparam )
```

Multi-threads parallel ILU decoposition of a BSR matrix A based on graph coloring.

Parameters

A	Pointer to dBSRmat matrix
Ap	Pointer to dCSRmat matrix and provide sparsity pattern
iludata	Pointer to ILU_data
iluparam	Pointer to ILU_param

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Zheng Li

Date

12/04/2016

Note

Only works for 1, 2, 3 nb (Zheng)

Definition at line 401 of file BlaILUSetupBSR.c.

9.17.2.4 fasp_ilu_dbsr_setup_omp()

```
SHORT fasp_ilu_dbsr_setup_omp (
            dBSRmat * A,
            ILU_data * iludata,
            ILU_param * iluparam )
```

Multi-threads parallel ILU decoposition of a BSR matrix A based on graph coloring.

Parameters

A	Pointer to dBSRmat matrix
iludata	Pointer to ILU_data
iluparam	Pointer to ILU_param

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Zheng Li

Date

12/04/2016

Note

Only works for 1, 2, 3 nb (Zheng)

Definition at line 164 of file BlaILUSetupBSR.c.

9.18 BlaILUSetupCSR.c File Reference

Setup incomplete LU decomposition for dCSRmat matrices.

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

Functions

- SHORT fasp_ilu_dcsr_setup (dCSRmat *A, ILU_data *iludata, ILU_param *iluparam)
  Get ILU decomposition of a CSR matrix A.

9.18.1 Detailed Description

Setup incomplete LU decomposition for dCSRmat matrices.

Note

This file contains Level-1 (Bla) functions. It requires: AuxTiming.c, BlaILU.c, BlaSparseCSR.c, and PreDataInit.c

9.18.2 Function Documentation

9.18.2.1 fasp_ilu_dcsr_setup()

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

Get ILU decomposition of a CSR matrix A.

Parameters

A	Pointer to dCSRmat matrix
iludata	Pointer to ILU_data
iluparam	Pointer to ILU_param

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Shiquan Zhang Xiaozhe Hu

Date

12/27/2009

Definition at line 38 of file BlaILUSetupCSR.c.

9.19 BlaILUSetupSTR.c File Reference

Setup incomplete LU decomposition for dSTRmat matrices.

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

Functions

- void fasp_ilu_dstr_setup0 (dSTRmat ∗A, dSTRmat ∗LU)

  Get ILU(0) decomposition of a structured matrix A.
- void fasp_ilu_dstr_setup1 (dSTRmat ∗A, dSTRmat ∗LU)

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

9.19.1 Detailed Description

Setup incomplete LU decomposition for dSTRmat matrices.

Note

This file contains Level-1 (Bla) functions. It requires: AuxMemory.c, BlaSmallMat.c, BlaSmallMatInv.c, Bla←SparseSTR.c, and BlaArray.c

9.19.2 Function Documentation

9.19.2.1 fasp_ilu_dstr_setup0()

```
void fasp_ilu_dstr_setup0 (
            dSTRmat * A,
            dSTRmat * LU )
```

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

Parameters

A	Pointer to dSTRmat
LU	Pointer to ILU structured matrix of REAL type

Author

Shiquan Zhang, Xiaozhe Hu

Date

11/08/2010

Note

Only works for 5 bands 2D and 7 bands 3D matrix with default offsets (order can be arbitrary)!

Definition at line 38 of file BlaILUSetupSTR.c.

9.19.2.2 fasp_ilu_dstr_setup1()

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

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

Parameters

A	Pointer to oringinal structured matrix of REAL type
LU	Pointer to ILU structured matrix of REAL type

Author

Shiquan Zhang, Xiaozhe Hu

Date

11/08/2010

Note

Put L and U in a STR matrix and it has the following structure: the diag is d, the offdiag of L are alpha1 to alpha6, the offdiag of U are beta1 to beta6
Only works for 5 bands 2D and 7 bands 3D matrix with default offsets

Definition at line 333 of file BlaILUSetupSTR.c.

9.20 BlaIO.c File Reference

Matrix/vector input/output subroutines.

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

Functions

- void fasp_dcsrvec1_read (const char ∗filename, dCSRmat ∗A, dvector ∗b)

  Read A and b from a SINGLE disk file.
- void fasp_dcsrvec2_read (const char ∗filemat, const char ∗filerhs, dCSRmat ∗A, dvector ∗b)

  Read A and b from two disk files.
- void fasp_dcsr_read (const char ∗filename, dCSRmat ∗A)

  Read A from matrix disk file in IJ format.
- void fasp_dcoo_read (const char ∗filename, dCSRmat ∗A)

  Read A from matrix disk file in IJ format – indices starting from 0.
- void fasp_dcoo1_read (const char ∗filename, dCOOmat ∗A)

  Read A from matrix disk file in IJ format – indices starting from 1.
- void fasp_dcoo_shift_read (const char ∗filename, dCSRmat ∗A)

  Read A from matrix disk file in IJ format – indices starting from 0.
- void fasp_dmtx_read (const char ∗filename, dCSRmat ∗A)

Read A from matrix disk file in MatrixMarket general format.
- void fasp_dmtxsym_read (const char ∗filename, dCSRmat ∗A)

  Read A from matrix disk file in MatrixMarket sym format.
- void fasp_dstr_read (const char ∗filename, dSTRmat ∗A)

  Read A from a disk file in dSTRmat format.
- void fasp_dbsr_read (const char ∗filename, dBSRmat ∗A)

  Read A from a disk file in dBSRmat format.
- void fasp_dvecind_read (const char ∗filename, dvector ∗b)

  Read b from matrix disk file.
- void fasp_dvec_read (const char ∗filename, dvector ∗b)

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

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

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

  Write A and b to a SINGLE disk file.
- void fasp_dcsrvec2_write (const char ∗filemat, const char ∗filerhs, dCSRmat ∗A, dvector ∗b)

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

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

  Write a dSTRmat to a disk file.
- void fasp_dbsr_write (const char ∗filename, dBSRmat ∗A)

  Write a dBSRmat to a disk file.
- void fasp_dvec_write (const char ∗filename, dvector ∗vec)

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

  Write a dvector to disk file in coordinate format.
- void fasp_ivec_write (const char ∗filename, ivector ∗vec)

  Write a ivector to disk file in coordinate format.
- void fasp_dvec_print (const INT n, dvector ∗u)

  Print first n entries of a vector of REAL type.
- void fasp_ivec_print (const INT n, ivector ∗u)

  Print first n entries of a vector of INT type.
- void fasp_dcsr_print (const dCSRmat ∗A)

  Print out a dCSRmat matrix in coordinate format.
- void fasp_dcoo_print (const dCOOmat ∗A)

  Print out a dCOOmat matrix in coordinate format.
- void fasp_dbsr_print (const dBSRmat ∗A)

  Print out a dBSRmat matrix in coordinate format.
- void fasp_dbsr_write_coo (const char ∗filename, const dBSRmat ∗A)

  Print out a dBSRmat matrix in coordinate format for matlab spy.
- void fasp_dcsr_write_coo (const char ∗filename, const dCSRmat ∗A)

  Print out a dCSRmat matrix in coordinate format for matlab spy.
- void fasp_dstr_print (const dSTRmat ∗A)

  Print out a dSTRmat matrix in coordinate format.

- void fasp_matrix_read (const char *filename, void *A)

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

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

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

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

    write RHS vector from different kinds of formats in both ASCII and binary files
- void fasp_hb_read (const char *input_file, dCSRmat *A, dvector *b)

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

Variables

- INT ilength
- INT dlength

9.20.1 Detailed Description

Matrix/vector input/output subroutines.

Note

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

This file contains Level-1 (Bla) functions. It requires: AuxArray.c, AuxConvert.c, AuxMemory.c, AuxMessage.c, AuxVector.c, BlaFormat.c, BlaSparseBSR.c, BlaSparseCOO.c, BlaSparseCSR.c, and BlaSpmvCSR.c

9.20.2 Function Documentation

9.20.2.1 fasp_dbsr_print()

```
void fasp_dbsr_print (
            const dBSRmat * A )
```

Print out a dBSRmat matrix in coordinate format.

Parameters

A	Pointer to the dBSRmat matrix A

Author

Ziteng Wang

Date

12/24/2012

Modified by Chunsheng Feng on 11/16/2013

Definition at line 1448 of file BlaIO.c.

9.20.2.2 fasp_dbsr_read()

```
void fasp_dbsr_read (
            const char * filename,
            dBSRmat * A )
```

Read A from a disk file in dBSRmat format.

Parameters

filename	File name for matrix A
A	Pointer to the dBSRmat A

Note

This routine reads a dBSRmat matrix from a disk file in the following format:
File format:

- ROW, COL, NNZ
- nb: size of each block
- storage_manner: storage manner of each block
- ROW+1: length of IA
- IA(i), i=0:ROW
- NNZ: length of JA
- JA(i), i=0:NNZ-1
- NNZ∗nb∗nb: length of val
- val(i), i=0:NNZ∗nb∗nb-1

Author

Xiaozhe Hu

Date

10/29/2010

Definition at line 700 of file BlaIO.c.

9.20.2.3 fasp_dbsr_write()

```
void fasp_dbsr_write (
            const char * filename,
            dBSRmat * A )
```

Write a dBSRmat to a disk file.

Parameters

filename	File name for A
A	Pointer to the dBSRmat matrix A

Note

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

Author

Shiquan Zhang

Date

10/29/2010

Definition at line 1201 of file BlaIO.c.

9.20.2.4 fasp_dbsr_write_coo()

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

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

Parameters

filename	Name of file to write to
A	Pointer to the dBSRmat matrix A

Author

Chunsheng Feng

Date

11/14/2013

Modified by Chensong Zhang on 06/14/2014: Fix index problem.

Definition at line 1484 of file BlaIO.c.

9.20.2.5 fasp_dcoo1_read()

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

Read A from matrix disk file in IJ format – indices starting from 1.

Parameters

filename	File name for matrix
A	Pointer to the COO matrix

Note

File format:

- nrow ncol nnz % number of rows, number of columns, and nnz
- i j a_ij % i, j a_ij in each line

Difference between fasp_dcoo_read and this funciton is this function do not change to CSR format

Author

Xiaozhe Hu

Date

03/24/2013

Definition at line 380 of file BlaIO.c.

9.20.2.6 fasp_dcoo_print()

```
void fasp_dcoo_print (
            const dCOOmat * A )
```

Print out a dCOOmat matrix in coordinate format.

Parameters

A	Pointer to the dCOOmat matrix A

Author

Ziteng Wang

Date

12/24/2012

Definition at line 1426 of file BlaIO.c.

9.20.2.7 fasp_dcoo_read()

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

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

Parameters

filename	File name for matrix
A	Pointer to the CSR matrix

Note

File format:

- nrow ncol nnz % number of rows, number of columns, and nnz
- i j a_ij % i, j a_ij in each line

After reading, it converts the matrix to dCSRmat format.

Author

Xuehai Huang, Chensong Zhang

Date

03/29/2009

Definition at line 328 of file BlaIO.c.

9.20.2.8 fasp_dcoo_shift_read()

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

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

Parameters

filename	File name for matrix
A	Pointer to the CSR matrix

Note

File format:

- nrow ncol nnz % number of rows, number of columns, and nnz
- i j a_ij % i, j a_ij in each line

i and j suppose to start with index 1!!!
After read in, it shifts the index to C fashin and converts the matrix to dCSRmat format.

Author

Xiaozhe Hu

Date

04/01/2014

Definition at line 431 of file BlaIO.c.

9.20.2.9 fasp_dcoo_write()

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

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

Parameters

A	pointer to the dCSRmat matrix
filename	char for vector file name

Note

The routine writes the specified REAL vector in COO format. Refer to the reading subroutine 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 1106 of file BlaIO.c.

9.20.2.10 fasp_dcsr_print()

```
void fasp_dcsr_print (
            const dCSRmat * A )
```

Print out a dCSRmat matrix in coordinate format.

Parameters

A	Pointer to the dCSRmat matrix A

Author

Xuehai Huang

Date

03/29/2009

Definition at line 1404 of file BlaIO.c.

9.20.2.11 fasp_dcsr_read()

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

Read A from matrix disk file in IJ format.

Parameters

filename	Char for matrix file name
A	Pointer to the CSR matrix

Author

Ziteng Wang

Date

12/25/2012

Definition at line 267 of file BlaIO.c.

9.20.2.12 fasp_dcsr_write_coo()

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

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

Parameters

filename	Name of file to write to
A	Pointer to the dCSRmat matrix A

Author

Chunsheng Feng

Date

11/14/2013

Definition at line 1535 of file BlaIO.c.

9.20.2.13 fasp_dcsrvec1_read()

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

Read A and b from a SINGLE disk file.

Parameters

filename	File name
A	Pointer to the CSR matrix
b	Pointer to the dvector

Note

This routine reads a dCSRmat matrix and a dvector vector from a single disk file. The difference between this and fasp_dcoovec_read is that this routine support non-square matrices.
File format:

- nrow ncol % number of rows and number of columns
- ia(j), j=0:nrow % row index
- ja(j), j=0:nnz-1 % column index
- a(j), j=0:nnz-1 % entry value
- n % number of entries
- b(j), j=0:n-1 % entry value

Author

Xuehai Huang

Date

03/29/2009

Modified by Chensong Zhang on 03/14/2012

Definition at line 99 of file BlaIO.c.

9.20.2.14 fasp_dcsrvec1_write()

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

Write A and b to a SINGLE disk file.

Parameters

filename	File name
A	Pointer to the CSR matrix
b	Pointer to the dvector

Note

This routine writes a dCSRmat matrix and a dvector vector to a single disk file.
File format:

- nrow ncol % number of rows and number of columns
- ia(j), j=0:nrow % row index
- ja(j), j=0:nnz-1 % column index
- a(j), j=0:nnz-1 % entry value
- n % number of entries
- b(j), j=0:n-1 % entry value

Author

Feiteng Huang

Date

05/19/2012

Modified by Chensong on 12/26/2012

Definition at line 959 of file BlaIO.c.

9.20.2.15 fasp_dcsrvec2_read()

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

Read A and b from two disk files.

Parameters

filemat	File name for matrix
filerhs	File name for right-hand side
A	Pointer to the dCSR matrix
b	Pointer to the dvector

Note

This routine reads a dCSRmat matrix and a dvector vector from a disk file.
CSR matrix file format:

- nrow % number of columns (rows)
- ia(j), j=0:nrow % row index
- ja(j), j=0:nnz-1 % column index
- a(j), j=0:nnz-1 % entry value

RHS file format:

- n % number of entries
- b(j), j=0:nrow-1 % entry value

Indices start from 1, NOT 0!!!

Author

Zhiyang Zhou

Date

2010/08/06

Modified by Chensong Zhang on 2012/01/05

Definition at line 188 of file BlaIO.c.

9.20.2.16 fasp_dcsrvec2_write()

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

Write A and b to two disk files.

Parameters

filemat	File name for matrix
filerhs	File name for right-hand side
A	Pointer to the dCSR matrix
b	Pointer to the dvector

Note

This routine writes a dCSRmat matrix and a dvector vector to two disk files.
CSR matrix file format:

- nrow % number of columns (rows)
- ia(j), j=0:nrow % row index
- ja(j), j=0:nnz-1 % column index
- a(j), j=0:nnz-1 % entry value

RHS file format:

- n % number of entries
- b(j), j=0:nrow-1 % entry value

Indices start from 1, NOT 0!!!

Author

Feiteng Huang

Date

05/19/2012

Definition at line 1036 of file BlaIO.c.

9.20.2.17 fasp_dmtx_read()

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

Read A from matrix disk file in MatrixMarket general format.

Parameters

filename	File name for matrix
A	Pointer to the CSR matrix

Note

File format: This routine reads a MatrixMarket general matrix from a mtx file. And it converts the matrix to dCS←
Rmat format. For details of mtx format, please refer to `http://math.nist.gov/MatrixMarket/`.
Indices start from 1, NOT 0!!!

Author

Chensong Zhang

Date

09/05/2011

Definition at line 483 of file BlaIO.c.

9.20.2.18 fasp_dmtxsym_read()

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

Read A from matrix disk file in MatrixMarket sym format.

Parameters

filename	File name for matrix
A	Pointer to the CSR matrix

Note

File format: This routine reads a MatrixMarket symmetric matrix from a mtx file. And it converts the matrix to dCSRmat format. For details of mtx format, please refer to `http://math.nist.gov/MatrixMarket/`. Indices start from 1, NOT 0!!!

Author

Chensong Zhang

Date

09/02/2011

Definition at line 543 of file BlaIO.c.

9.20.2.19 fasp_dstr_print()

```
void fasp_dstr_print (
            const dSTRmat * A )
```

Print out a dSTRmat matrix in coordinate format.

Parameters

A	Pointer to the dSTRmat matrix A

Author

Ziteng Wang

Date

12/24/2012

Definition at line 1574 of file BlaIO.c.

9.20.2.20 fasp_dstr_read()

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

Read A from a disk file in dSTRmat format.

Parameters

filename	File name for the matrix
A	Pointer to the dSTRmat

Note

This routine reads a dSTRmat matrix from a disk file. After done, it converts the matrix to dCSRmat format.
File format:

- nx, ny, nz
- nc: number of components
- nband: number of bands
- n: size of diagonal, you must have diagonal
- diag(j), j=0:n-1
- offset, length: offset and length of off-diag1
- offdiag(j), j=0:length-1

Author

Xuehai Huang

Date

03/29/2009

Definition at line 620 of file BlaIO.c.

9.20.2.21 fasp_dstr_write()

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

Write a dSTRmat to a disk file.

Parameters

filename	File name for A
A	Pointer to the dSTRmat matrix A

Note

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

Author

Shiquan Zhang

Date

03/29/2010

Definition at line 1144 of file BlaIO.c.

9.20.2.22 fasp_dvec_print()

```
void fasp_dvec_print (
            const INT n,
            dvector * u )
```

Print first n entries of a vector of REAL type.

Parameters

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

Author

Chensong Zhang

Date

03/29/2009

Definition at line 1361 of file BlaIO.c.

9.20.2.23 fasp_dvec_read()

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

Read b from a disk file in array format.

Parameters

filename	File name for vector b
b	Pointer to the dvector b (output)

Note

File Format:

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

Author

Chensong Zhang

Date

03/29/2009

Definition at line 819 of file BlaIO.c.

9.20.2.24 fasp_dvec_write()

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

Write a dvector to disk file.

Parameters

vec	Pointer to the dvector
filename	File name

Author

Xuehai Huang

Date

03/29/2009

Definition at line 1256 of file BlaIO.c.

9.20.2.25 fasp_dvecind_read()

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

Read b from matrix disk file.

Parameters

filename	File name for vector b
b	Pointer to the dvector b (output)

Note

File Format:

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

Because the index is given, order is not important!

Author

Chensong Zhang

Date

03/29/2009

Definition at line 769 of file BlaIO.c.

9.20.2.26 fasp_dvecind_write()

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

Write a dvector to disk file in coordinate format.

Parameters

vec	Pointer to the dvector
filename	File name

Note

The routine writes the specified REAL vector in IJ format.

- The first line of the file is the length of the vector;
- After that, each line gives index and value of the entries.

Author

Xuehai Huang

Date

03/29/2009

Definition at line 1292 of file BlaIO.c.

9.20.2.27 fasp_hb_read()

```
fasp_hb_read (
            const char * input_file,
            dCSRmat * A,
            dvector * b )
```

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

Parameters

input_file	File name of vector file
A	Pointer to the matrix
b	Pointer to the vector

Note

Modified from the c code hb_io_prb.c by John Burkardt

Author

Xiaoehe Hu

Date

05/30/2014

Definition at line 2064 of file BlaIO.c.

9.20.2.28 fasp_ivec_print()

```
void fasp_ivec_print (
            const INT n,
            ivector * u )
```

Print first n entries of a vector of INT type.

Parameters

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

Author

Chensong Zhang

Date

03/29/2009

Definition at line 1383 of file BlaIO.c.

9.20.2.29 fasp_ivec_read()

```
void fasp_ivec_read (
            const char * filename,
            ivector * b )
```

Read b from a disk file in array format.

Parameters

filename	File name for vector b
b	Pointer to the dvector b (output)

Note

File Format:

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

Author

Xuehai Huang

Date

03/29/2009

Definition at line 909 of file BlaIO.c.

9.20.2.30 fasp_ivec_write()

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

Write a ivector to disk file in coordinate format.

Parameters

vec	Pointer to the dvector
filename	File name

Note

The routine writes the specified INT vector in IJ format.

- The first line of the file is the length of the vector;
- After that, each line gives index and value of the entries.

Author

Xuehai Huang

Date

03/29/2009

Definition at line 1327 of file BlaIO.c.

9.20.2.31 fasp_ivecind_read()

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

Read b from matrix disk file.

Parameters

filename	File name for vector b
b	Pointer to the dvector b (output)

Note

File Format:

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

Author

Chensong Zhang

Date

03/29/2009

Definition at line 869 of file BlaIO.c.

9.20.2.32 fasp_matrix_read()

```
fasp_matrix_read (
            const char * filename,
            void * A )
```

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

Parameters

filename	File name of matrix file
A	Pointer to the matrix

Note

Flags for matrix file format:

- fileflag % fileflag = 1: binary, fileflag = 0000: ASCII
- formatflag % a 3-digit number for internal use, see below
- matrix % different types of matrix

Meaning of formatflag:

- matrixflag % first digit of formatflag
  - matrixflag = 1: CSR format
  - matrixflag = 2: BSR format
  - matrixflag = 3: STR format
  - matrixflag = 4: COO format
  - matrixflag = 5: MTX format
  - matrixflag = 6: MTX symmetrical format
- ilength % third digit of formatflag, length of INT
- dlength % fourth digit of formatflag, length of REAL

Author

Ziteng Wang

Date

12/24/2012

Modified by Chensong Zhang on 05/01/2013

Definition at line 1608 of file BlaIO.c.

9.20.2.33 fasp_matrix_read_bin()

```
void fasp_matrix_read_bin (
            const char * filename,
            void * A )
```

Read matrix in binary format.

Parameters

filename	File name of matrix file
A	Pointer to the matrix

Author

Xiaozhe Hu

Date

04/14/2013

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

Definition at line 1709 of file BlaIO.c.

9.20.2.34 fasp_matrix_write()

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

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

Parameters

filename	File name of matrix file
A	Pointer to the matrix
flag	Type of file and matrix, a 3-digit number

Note

Meaning of flag:

- fileflag % fileflag = 1: binary, fileflag = 0: ASCII
- matrixflag
  - matrixflag = 1: CSR format
  - matrixflag = 2: BSR format
  - matrixflag = 3: STR format

Matrix file format:

- fileflag % fileflag = 1: binary, fileflag = 0000: ASCII
- formatflag % a 3-digit number
- matrixflag % different kinds of matrix judged by formatflag

Author

Ziteng Wang

Date

12/24/2012

Definition at line 1783 of file BlaIO.c.

9.20.2.35 fasp_vector_read()

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

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

Parameters

filerhs	File name of vector file
b	Pointer to the vector

Note

Matrix file format:

- fileflag % fileflag = 1: binary, fileflag = 0000: ASCII
- formatflag % a 3-digit number
- vector % different kinds of vector judged by formatflag

Meaning of formatflag:

- vectorflag % first digit of formatflag
  - vectorflag = 1: dvec format
  - vectorflag = 2: ivec format
  - vectorflag = 3: dvecind format
  - vectorflag = 4: ivecind format
- ilength % second digit of formatflag, length of INT
- dlength % third digit of formatflag, length of REAL

Author

Ziteng Wang

Date

12/24/2012

Definition at line 1877 of file BlaIO.c.

9.20.2.36 fasp_vector_write()

```
fasp_vector_write (
            const char * filerhs,
            void * b,
            const INT flag )
```

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

Parameters

filerhs	File name of vector file
b	Pointer to the vector
flag	Type of file and vector, a 2-digit number

Note

Meaning of the flags

- fileflag % fileflag = 1: binary, fileflag = 0: ASCII
- vectorflag
  - vectorflag = 1: dvec format
  - vectorflag = 2: ivec format
  - vectorflag = 3: dvecind format
  - vectorflag = 4: ivecind format

Matrix file format:

- fileflag % fileflag = 1: binary, fileflag = 0000: ASCII
- formatflag % a 2-digit number
- vectorflag % different kinds of vector judged by formatflag

Author

Ziteng Wang

Date

12/24/2012

Modified by Chensong Zhang on 05/02/2013: fix a bug when writing in binary format

Definition at line 1975 of file BlaIO.c.

9.20.3 Variable Documentation

9.20.3.1 dlength

`INT dlength`

Length of REAL in byte

Definition at line 24 of file BlaIO.c.

9.20.3.2 ilength

`INT ilength`

Length of INT in byte

Definition at line 23 of file BlaIO.c.

9.21 BlaOrderingCSR.c File Reference

Generating ordering using algebraic information.

`#include "fasp.h"`

Functions

- void fasp_dcsr_CMK_order (const dCSRmat ∗A, INT ∗order, INT ∗oindex)
  Ordering vertices of matrix graph corresponding to A.
- void fasp_dcsr_RCMK_order (const dCSRmat ∗A, INT ∗order, INT ∗oindex, INT ∗rorder)
  Resverse CMK ordering.

9.21.1 Detailed Description

Generating ordering using algebraic information.

Note

This file contains Level-1 (Bla) functions.

9.21.2 Function Documentation

9.21.2.1 fasp_dcsr_CMK_order()

```
void fasp_dcsr_CMK_order (
            const dCSRmat * A,
            INT * order,
            INT * oindex )
```

Ordering vertices of matrix graph corresponding to A.

Parameters

A	Pointer to matrix
oindex	Pointer to index of vertices in order
order	Pointer to vertices with increasing degree

Author

Zheng Li, Chensong Zhang

Date

05/28/2014

Definition at line 37 of file BlaOrderingCSR.c.

9.21.2.2 fasp_dcsr_RCMK_order()

```
void fasp_dcsr_RCMK_order (
            const dCSRmat * A,
            INT * order,
            INT * oindex,
            INT * rorder )
```

Resverse CMK ordering.

Parameters

A	Pointer to matrix
order	Pointer to vertices with increasing degree
oindex	Pointer to index of vertices in order
rorder	Pointer to reverse order

Author

Zheng Li, Chensong Zhang

Date

10/10/2014

Definition at line 87 of file BlaOrderingCSR.c.

9.22 BlaSchwarzSetup.c File Reference

Setup phase for the Schwarz methods.

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

Functions

- INT fasp_swz_dcsr_setup (SWZ_data *swzdata, SWZ_param *swzparam)

  Setup phase for the Schwarz methods.
- void fasp_dcsr_swz_forward_smoother (SWZ_data *swzdata, SWZ_param *swzparam, dvector *x, dvector *b)

  Schwarz smoother: forward sweep.
- void fasp_dcsr_swz_backward_smoother (SWZ_data *swzdata, SWZ_param *swzparam, dvector *x, dvector *b)

  Schwarz smoother: backward sweep.

9.22.1 Detailed Description

Setup phase for the Schwarz methods.

Note

This file contains Level-1 (Bla) functions. It requires: AuxMemory.c, AuxVector.c, BlaSparseCSR.c, BlaSparseUtil.c, and KryPvgmres.c

9.22.2 Function Documentation

9.22.2.1 fasp_dcsr_swz_backward_smoother()

```
void fasp_dcsr_swz_backward_smoother (
            SWZ_data * swzdata,
            SWZ_param * swzparam,
            dvector * x,
            dvector * b )
```

Schwarz smoother: backward sweep.

Parameters

swzdata	Pointer to the Schwarz data
swzparam	Pointer to the Schwarz parameter
x	Pointer to solution vector
b	Pointer to right hand

Author

Zheng Li, Chensong Zhang

Date

2014/10/5

Definition at line 325 of file BlaSchwarzSetup.c.

9.22.2.2 fasp_dcsr_swz_forward_smoother()

```
void fasp_dcsr_swz_forward_smoother (
            SWZ_data * swzdata,
            SWZ_param * swzparam,
            dvector * x,
            dvector * b )
```

Schwarz smoother: forward sweep.

Parameters

swzdata	Pointer to the Schwarz data
swzparam	Pointer to the Schwarz parameter
x	Pointer to solution vector
b	Pointer to right hand

Author

Zheng Li, Chensong Zhang

Date

2014/10/5

Definition at line 215 of file BlaSchwarzSetup.c.

9.22.2.3 fasp_swz_dcsr_setup()

```
INT fasp_swz_dcsr_setup (
            SWZ_data * swzdata,
            SWZ_param * swzparam )
```

Setup phase for the Schwarz methods.

Parameters

swzdata	Pointer to the Schwarz data
swzparam	Type of the Schwarz method

Returns

FASP_SUCCESS if succeed

Author

Ludmil, Xiaozhe Hu

Date

03/22/2011

Modified by Zheng Li on 10/09/2014

Definition at line 47 of file BlaSchwarzSetup.c.

9.23 BlaSmallMat.c File Reference

BLAS operations for *small* dense matrices.

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

Functions

- void fasp_blas_smat_axm (REAL *a, const INT n, const REAL alpha)
  *Compute a = alpha*a (in place)*
- void fasp_blas_smat_add (const REAL *a, const REAL *b, const INT n, const REAL alpha, const REAL beta, REAL *c)
  *Compute c = alpha*a + beta*b.*
- void fasp_blas_smat_mxv_nc2 (const REAL *a, const REAL *b, REAL *c)
  *Compute the product of a 2*2 matrix a and a array b, stored in c.*
- void fasp_blas_smat_mxv_nc3 (const REAL *a, const REAL *b, REAL *c)
  *Compute the product of a 3*3 matrix a and a array b, stored in c.*
- void fasp_blas_smat_mxv_nc5 (const REAL *a, const REAL *b, REAL *c)
  *Compute the product of a 5*5 matrix a and a array b, stored in c.*
- void fasp_blas_smat_mxv_nc7 (const REAL *a, const REAL *b, REAL *c)
  *Compute the product of a 7*7 matrix a and a array b, stored in c.*
- void fasp_blas_smat_mxv (const REAL *a, const 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 (const REAL *a, const REAL *b, REAL *c)
  Compute the matrix product of two 2 matrices a and b, stored in c.*
- void fasp_blas_smat_mul_nc3 (const REAL *a, const 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 (const REAL *a, const 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 (const REAL *a, const REAL *b, REAL *c)
  *Compute the matrix product of two 7*7 matrices a and b, stored in c.*
- void fasp_blas_smat_mul (const REAL *a, const 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_smat_ypAx_nc2 (const REAL *A, const REAL *x, REAL *y)
  *Compute y := y + Ax, where 'A' is a 2*2 dense matrix.*
- void fasp_blas_smat_ypAx_nc3 (const REAL *A, const REAL *x, REAL *y)
  *Compute y := y + Ax, where 'A' is a 3*3 dense matrix.*
- void fasp_blas_smat_ypAx_nc5 (const REAL *A, const REAL *x, REAL *y)
  *Compute y := y + Ax, where 'A' is a 5*5 dense matrix.*
- void fasp_blas_smat_ypAx_nc7 (const REAL *A, const REAL *x, REAL *y)
  *Compute y := y + Ax, where 'A' is a 7*7 dense matrix.*
- void fasp_blas_smat_ypAx (const REAL *A, const 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 (const REAL *A, const REAL *x, REAL *y)
  *Compute y := y - Ax, where 'A' is a 2*2 dense matrix.*
- void fasp_blas_smat_ymAx_nc3 (const REAL *A, const REAL *x, REAL *y)
  *Compute y := y - Ax, where 'A' is a 3*3 dense matrix.*
- void fasp_blas_smat_ymAx_nc5 (const REAL *A, const REAL *x, REAL *y)
  *Compute y := y - Ax, where 'A' is a 5*5 dense matrix.*
- void fasp_blas_smat_ymAx_nc7 (const REAL *A, const REAL *x, REAL *y)
  *Compute y := y - Ax, where 'A' is a 7*7 dense matrix.*
- void fasp_blas_smat_ymAx (const REAL *A, const REAL *x, REAL *y, const INT n)
  *Compute y := y - Ax, where 'A' is a n*n dense matrix.*
- void fasp_blas_smat_aAxpby (const REAL alpha, const REAL *A, const REAL *x, const REAL beta, REAL *y, const INT n)
  *Compute y:=alpha*A*x + beta*y.*

9.23.1 Detailed Description

BLAS operations for *small* dense matrices.

Note

This file contains Level-1 (Bla) functions. It requires: BlaSparseBSR.c, BlaSparseCSR.c, BlaSpmvCSR.c, and PreDataInit.c

Warning

These rountines are designed for full matrices only!
This file contains very long lines. Not print friendly!

9.23.2 Function Documentation

9.23.2.1 fasp_blas_smat_aAxpby()

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

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

Parameters

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

Author

Zhiyang Zhou, Chensong Zhang

Date

2010/10/25

Definition at line 930 of file BlaSmallMat.c.

9.23.2.2 fasp_blas_smat_add()

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

Compute c = alpha∗a + beta∗b.

Parameters

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

Author

Xiaozhe Hu, Chensong Zhang

Date

05/26/2014

Definition at line 65 of file BlaSmallMat.c.

9.23.2.3 fasp_blas_smat_axm()

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

Compute a = alpha∗a (in place)

Parameters

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

Author

Xiaozhe Hu, Chensong Zhang

Date

05/26/2014

Definition at line 37 of file BlaSmallMat.c.

9.23.2.4 fasp_blas_smat_mul()

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

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

Parameters

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

Author

Xiaozhe Hu, Shiquan Zhang

Date

04/21/2010

Definition at line 458 of file BlaSmallMat.c.

9.23.2.5 fasp_blas_smat_mul_nc2()

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

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

Parameters

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

Author

Xiaozhe Hu

Date

18/11/2011

Definition at line 243 of file BlaSmallMat.c.

9.23.2.6 fasp_blas_smat_mul_nc3()

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

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

Parameters

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

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 272 of file BlaSmallMat.c.

9.23.2.7 fasp_blas_smat_mul_nc5()

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

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

Parameters

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

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 309 of file BlaSmallMat.c.

9.23.2.8 fasp_blas_smat_mul_nc7()

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

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

Parameters

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

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 368 of file BlaSmallMat.c.

9.23.2.9 fasp_blas_smat_mxv()

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

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

Parameters

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

Author

Xiaozhe Hu, Shiquan Zhang

Date

04/21/2010

Definition at line 193 of file BlaSmallMat.c.

9.23.2.10 fasp_blas_smat_mxv_nc2()

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

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

Parameters

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

Author

Xiaozhe Hu

Date

18/11/2010

Definition at line 93 of file BlaSmallMat.c.

9.23.2.11 fasp_blas_smat_mxv_nc3()

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

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

Parameters

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

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 115 of file BlaSmallMat.c.

9.23.2.12 fasp_blas_smat_mxv_nc5()

```
void fasp_blas_smat_mxv_nc5 (
            const REAL * a,
            const REAL * b,
            REAL * c )
```

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

Parameters

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

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 138 of file BlaSmallMat.c.

9.23.2.13 fasp_blas_smat_mxv_nc7()

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

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

Parameters

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

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 164 of file BlaSmallMat.c.

9.23.2.14 fasp_blas_smat_ymAx()

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

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

Parameters

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

Author

Zhiyang Zhou, Xiaozhe Hu, Chensong Zhang

Date

2010/10/25

Modified by Chensong Zhang on 01/25/2017

Definition at line 828 of file BlaSmallMat.c.

9.23.2.15 fasp_blas_smat_ymAx_nc2()

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

Compute y := y - Ax, where 'A' is a 2*2 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 713 of file BlaSmallMat.c.

9.23.2.16 fasp_blas_smat_ymAx_nc3()

```
void fasp_blas_smat_ymAx_nc3 (
            const REAL * A,
            const 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

Xiaozhe Hu, Zhiyang Zhou

Date

01/06/2011

Note

Works for 3-component

Definition at line 739 of file BlaSmallMat.c.

9.23.2.17 fasp_blas_smat_ymAx_nc5()

```
void fasp_blas_smat_ymAx_nc5 (
            const REAL * A,
            const 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

Xiaozhe Hu, Zhiyang Zhou

Date

01/06/2011

Note

Works for 5-component

Definition at line 766 of file BlaSmallMat.c.

9.23.2.18 fasp_blas_smat_ymAx_nc7()

```
void fasp_blas_smat_ymAx_nc7 (
            const REAL * A,
            const 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 795 of file BlaSmallMat.c.

9.23.2.19 fasp_blas_smat_ypAx()

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

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

Parameters

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

Author

Zhiyang Zhou, Chensong Zhang

Date

2010/10/25

Modified by Chensong Zhang on 01/25/2017

Definition at line 613 of file BlaSmallMat.c.

9.23.2.20 fasp_blas_smat_ypAx_nc2()

```
void fasp_blas_smat_ypAx_nc2 (
            const REAL * A,
            const 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 507 of file BlaSmallMat.c.

9.23.2.21 fasp_blas_smat_ypAx_nc3()

```
void fasp_blas_smat_ypAx_nc3 (
            const REAL * A,
            const 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 531 of file BlaSmallMat.c.

9.23.2.22 fasp_blas_smat_ypAx_nc5()

```
void fasp_blas_smat_ypAx_nc5 (
            const REAL * A,
            const 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, Chensong Zhang

Date

2010/10/25

Definition at line 555 of file BlaSmallMat.c.

9.23.2.23 fasp_blas_smat_ypAx_nc7()

```
void fasp_blas_smat_ypAx_nc7 (
            const REAL * A,
            const 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, Chensong Zhang

Date

2010/10/25

Definition at line 581 of file BlaSmallMat.c.

9.24 BlaSmallMatInv.c File Reference

Find inversion of *small* dense matrices in row-major format.

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

Macros

- #define SWAP(a, b) {temp=(a);(a)=(b);(b)=temp;}

Functions

- void fasp_smat_inv_nc2 (REAL ∗a)

  Compute the inverse matrix of a 2∗2 full matrix A (in place)
- void fasp_smat_inv_nc3 (REAL ∗a)

  Compute the inverse matrix of a 3∗3 full matrix A (in place)
- void fasp_smat_inv_nc4 (REAL ∗a)

  Compute the inverse matrix of a 4∗4 full matrix A (in place)
- void fasp_smat_inv_nc5 (REAL ∗a)

  Compute the inverse matrix of a 5∗5 full matrix A (in place)
- void fasp_smat_inv_nc7 (REAL ∗a)

  Compute the inverse matrix of a 7∗7 matrix a.
- void fasp_smat_inv_nc (REAL ∗a, const INT n)

  Compute the inverse of a matrix using Gauss Elimination.
- void fasp_smat_invp_nc (REAL ∗a, const INT n)

  Compute the inverse of a matrix using Gauss Elimination with Pivoting.
- INT fasp_smat_inv (REAL ∗a, const INT n)

  Compute the inverse matrix of a small full matrix a.
- REAL fasp_smat_Linf (const REAL ∗A, const INT n)

  Compute the L infinity norm of A.
- void fasp_smat_identity_nc2 (REAL ∗a)

  Set a 2∗2 full matrix to be a identity.
- void fasp_smat_identity_nc3 (REAL ∗a)

  Set a 3∗3 full matrix to be a identity.
- void fasp_smat_identity_nc5 (REAL ∗a)

  Set a 5∗5 full matrix to be a identity.
- void fasp_smat_identity_nc7 (REAL ∗a)

  Set a 7∗7 full matrix to be a identity.
- void fasp_smat_identity (REAL ∗a, const INT n, const INT n2)

  Set a n∗n full matrix to be a identity.

9.24.1 Detailed Description

Find inversion of *small* dense matrices in row-major format.

Note

This file contains Level-1 (Bla) functions. It requires: AuxMemory.c

9.24.2 Macro Definition Documentation

9.24.2.1 SWAP

```
#define SWAP(
                a,
                b ) {temp=(a);(a)=(b);(b)=temp;}
```

swap two numbers

Definition at line 17 of file BlaSmallMatInv.c.

9.24.3 Function Documentation

9.24.3.1 fasp_smat_identity()

```
void fasp_smat_identity (
            REAL * a,
            const INT n,
            const INT n2 )
```

Set a n∗n full matrix to be a identity.

Parameters

a	Pointer to the REAL vector which stands for a n∗n full matrix
n	Size of full matrix
n2	Length of the REAL vector which stores the n∗n full matrix

Author

Xiaozhe Hu

Date

2010/12/25

Definition at line 713 of file BlaSmallMatInv.c.

9.24.3.2 fasp_smat_identity_nc2()

```
void fasp_smat_identity_nc2 (
            REAL * a )
```

Set a 2∗2 full matrix to be a identity.

Parameters

a	Pointer to the REAL vector which stands for a 2∗2 full matrix

Author

Xiaozhe Hu

Date

2011/11/18

Definition at line 633 of file BlaSmallMatInv.c.

9.24.3.3 fasp_smat_identity_nc3()

```
void fasp_smat_identity_nc3 (
            REAL * a )
```

Set a 3∗3 full matrix to be a identity.

Parameters

a	Pointer to the REAL vector which stands for a 3∗3 full matrix

Author

Xiaozhe Hu

Date

2010/12/25

Definition at line 650 of file BlaSmallMatInv.c.

9.24.3.4 fasp_smat_identity_nc5()

```
void fasp_smat_identity_nc5 (
            REAL * a )
```

Set a 5∗5 full matrix to be a identity.

Parameters

a	Pointer to the REAL vector which stands for a 5*5 full matrix

Author

Xiaozhe Hu

Date

2010/12/25

Definition at line 667 of file BlaSmallMatInv.c.

9.24.3.5 fasp_smat_identity_nc7()

```
void fasp_smat_identity_nc7 (
            REAL * a )
```

Set a 7*7 full matrix to be a identity.

Parameters

a	Pointer to the REAL vector which stands for a 7*7 full matrix

Author

Xiaozhe Hu

Date

2010/12/25

Definition at line 688 of file BlaSmallMatInv.c.

9.24.3.6 fasp_smat_inv()

```
INT fasp_smat_inv (
            REAL * a,
            const INT n )
```

Compute the inverse matrix of a small full matrix a.

Parameters

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

Author

Xiaozhe Hu, Shiquan Zhang

Date

04/21/2010

Definition at line 564 of file BlaSmallMatInv.c.

9.24.3.7 fasp_smat_inv_nc()

```
void fasp_smat_inv_nc (
            REAL * a,
            const INT n )
```

Compute the inverse of a matrix using Gauss Elimination.

Parameters

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

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 415 of file BlaSmallMatInv.c.

9.24.3.8 fasp_smat_inv_nc2()

```
void fasp_smat_inv_nc2 (
            REAL * a )
```

Compute the inverse matrix of a 2∗2 full matrix A (in place)

Parameters

a	Pointer to the REAL array which stands a 2∗2 matrix

Author

Xiaozhe Hu

Date

18/11/2011

Definition at line 33 of file BlaSmallMatInv.c.

9.24.3.9 fasp_smat_inv_nc3()

```
void fasp_smat_inv_nc3 (
            REAL * a )
```

Compute the inverse matrix of a 3∗3 full matrix A (in place)

Parameters

a	Pointer to the REAL array which stands a 3∗3 matrix

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 69 of file BlaSmallMatInv.c.

9.24.3.10 fasp_smat_inv_nc4()

```
void fasp_smat_inv_nc4 (
            REAL * a )
```

Compute the inverse matrix of a 4∗4 full matrix A (in place)

Parameters

a	Pointer to the REAL array which stands a 4∗4 matrix

Author

Xiaozhe Hu

Date

01/12/2013

Modified by Hongxuan Zhang on 06/13/2014: Fix a bug in M23.

Definition at line 124 of file BlaSmallMatInv.c.

9.24.3.11 fasp_smat_inv_nc5()

```
void fasp_smat_inv_nc5 (
            REAL * a )
```

Compute the inverse matrix of a 5∗5 full matrix A (in place)

Parameters

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

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 183 of file BlaSmallMatInv.c.

9.24.3.12 fasp_smat_inv_nc7()

```
void fasp_smat_inv_nc7 (
            REAL * a )
```

Compute the inverse matrix of a 7∗7 matrix a.

Parameters

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

Note

This is NOT implemented yet!

Author

Xiaozhe Hu, Shiquan Zhang

Date

05/01/2010

Definition at line 399 of file BlaSmallMatInv.c.

9.24.3.13 fasp_smat_invp_nc()

```
void fasp_smat_invp_nc (
            REAL * a,
            const INT n )
```

Compute the inverse of a matrix using Gauss Elimination with Pivoting.

Parameters

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

Author

Chensong Zhang

Date

04/03/2015

Note

This routine is based on gaussj() from "Numerical Recipies in C"!

Definition at line 482 of file BlaSmallMatInv.c.

9.24.3.14 fasp_smat_Linf()

```
REAL fasp_smat_Linf (
            const REAL * A,
            const INT n )
```

Compute the L infinity norm of A.

Parameters

A	Pointer to the n∗n dense matrix
n	the dimension of the dense matrix

Author

Xiaozhe Hu

Date

05/26/2014

Definition at line 605 of file BlaSmallMatInv.c.

9.25 BlaSmallMatLU.c File Reference

LU decomposition and direct solver for small dense matrices.

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

Functions

- SHORT fasp_smat_lu_decomp (REAL ∗A, INT pivot[ ], const INT n)

  LU decomposition of A using Doolittle's method.
- SHORT fasp_smat_lu_solve (const REAL ∗A, REAL b[ ], const INT pivot[ ], REAL x[ ], const INT n)

  Solving Ax=b using LU decomposition.

9.25.1 Detailed Description

LU decomposition and direct solver for small dense matrices.

Note

This file contains Level-1 (Bla) functions.

9.25.2 Function Documentation

9.25.2.1 fasp_smat_lu_decomp()

```
SHORT fasp_smat_lu_decomp (
            REAL * A,
            INT pivot[],
            const INT n )
```

LU decomposition of A using Doolittle's method.

Parameters

A	Pointer to the full matrix
pivot	Pivoting positions
n	Size of matrix A

Returns

FASP_SUCCESS if successed; otherwise, error information.

Note

Use Doolittle's method to decompose the n x n matrix A into a unit lower triangular matrix L and an upper triangular matrix U such that A = LU. The matrices L and U replace the matrix A. The diagonal elements of L are 1 and are not stored.

The Doolittle method with partial pivoting is: Determine the pivot row and interchange the current row with the pivot row, then assuming that row k is the current row, k = 0, ..., n - 1 evaluate in order the following pair of expressions U[k][j] = A[k][j] - (L[k][0]∗U[0][j] + ... + L[k][k-1]∗U[k-1][j]) for j = k, k+1, ... , n-1 L[i][k] = (A[i][k] - (L[i][0]∗U[0][k] + . + L[i][k-1]∗U[k-1][k])) / U[k][k] for i = k+1, ... , n-1.

Author

Xuehai Huang

Date

04/02/2009

Definition at line 52 of file BlaSmallMatLU.c.

9.25.2.2 fasp_smat_lu_solve()

```
SHORT fasp_smat_lu_solve (
            const REAL * A,
            REAL b[],
            const INT pivot[],
            REAL x[],
            const INT n )
```

Solving Ax=b using LU decomposition.

Parameters

A	Pointer to the full matrix
b	Right hand side array (b is used as the working array!!!)
pivot	Pivoting positions
x	Pointer to the solution array
n	Size of matrix A

Returns

FASP_SUCCESS if successed; otherwise, error information.

Note

This routine uses Doolittle's method to solve the linear equation Ax = b. This routine is called after the matrix A has been decomposed into a product of a unit lower triangular matrix L and an upper triangular matrix U with pivoting. The solution proceeds by solving the linear equation Ly = b for y and subsequently solving the linear equation Ux = y for x.

Author

Xuehai Huang

Date

04/02/2009

Definition at line 124 of file BlaSmallMatLU.c.

9.26 BlaSparseBLC.c File Reference

Sparse matrix block operations.

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

Functions

- void fasp_dblc_free (dBLCmat *A)

    Free block CSR sparse matrix data memory space.

9.26.1 Detailed Description

Sparse matrix block operations.

Note

This file contains Level-1 (Bla) functions. It requires: AuxMemory.c and BlaSparseCSR.c

9.26.2 Function Documentation

9.26.2.1 fasp_dblc_free()

```
void fasp_dblc_free (
            dBLCmat * A )
```

Free block CSR sparse matrix data memory space.

Parameters

A	Pointer to the dBLCmat matrix

Author

Xiaozhe Hu

Date

04/18/2014

Definition at line 38 of file BlaSparseBLC.c.

9.27 BlaSparseBSR.c File Reference

Sparse matrix operations for dBSRmat matrices.

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

Functions

- dBSRmat fasp_dbsr_create (const INT ROW, const INT COL, const INT NNZ, const INT nb, const INT storage←_manner)

  Create BSR sparse matrix data memory space.
- void fasp_dbsr_alloc (const INT ROW, const INT COL, const INT NNZ, const INT nb, const INT storage_manner, dBSRmat ∗A)

  Allocate memory space for BSR format sparse matrix.
- void fasp_dbsr_free (dBSRmat ∗A)

  Free memory space for BSR format sparse matrix.
- void fasp_dbsr_cp (const dBSRmat ∗A, dBSRmat ∗B)

  copy a dCSRmat to a new one B=A
- INT fasp_dbsr_trans (const dBSRmat ∗A, dBSRmat ∗AT)

  Find A^T from given dBSRmat matrix A.
- SHORT fasp_dbsr_getblk (const dBSRmat ∗A, const INT ∗Is, const INT ∗Js, const INT m, const INT n, dBSRmat ∗B)

  Get a sub BSR matrix of A with specified rows and columns.
- SHORT fasp_dbsr_diagpref (dBSRmat ∗A)

  Reorder the column and data arrays of a square BSR matrix, so that the first entry in each row is the diagonal one.
- dvector fasp_dbsr_getdiaginv (const dBSRmat ∗A)

  Get D^{-1} of matrix A.
- dBSRmat fasp_dbsr_diaginv (const dBSRmat ∗A)

  Compute B := D^{-1}∗A, where 'D' is the block diagonal part of A.
- dBSRmat fasp_dbsr_diaginv2 (const dBSRmat ∗A, REAL ∗diaginv)

  Compute B := D^{-1}∗A, where 'D' is the block diagonal part of A.
- dBSRmat fasp_dbsr_diaginv3 (const dBSRmat ∗A, REAL ∗diaginv)

  Compute B := D^{-1}∗A, where 'D' is the block diagonal part of A.
- dBSRmat fasp_dbsr_diaginv4 (const dBSRmat ∗A, REAL ∗diaginv)

  Compute B := D^{-1}∗A, where 'D' is the block diagonal part of A.
- void fasp_dbsr_getdiag (INT n, const dBSRmat ∗A, REAL ∗diag)

  Abstract the diagonal blocks of a BSR matrix.
- dBSRmat fasp_dbsr_diagLU (const dBSRmat ∗A, REAL ∗DL, REAL ∗DU)

  Compute B := DL∗A∗DU. We decompose each diagonal block of A into LDU form and DL = diag(L^{-1}) and DU = diag(U^{-1}).
- dBSRmat fasp_dbsr_diagLU2 (dBSRmat ∗A, REAL ∗DL, REAL ∗DU)

  Compute B := DL∗A∗DU. We decompose each diagonal block of A into LDU form and DL = diag(L^{-1}) and DU = diag(U^{-1}).
- dBSRmat fasp_dbsr_perm (const dBSRmat ∗A, const INT ∗P)

  Apply permutation of A, i.e. Aperm=PAP' by the orders given in P.

9.27.1 Detailed Description

Sparse matrix operations for dBSRmat matrices.

Note

This file contains Level-1 (Bla) functions. It requires: AuxArray.c, AuxMemory.c, AuxThreads.c, BlaSmallMat.c, and BlaSmallMatInv.c

9.27.2 Function Documentation

9.27.2.1 fasp_dbsr_alloc()

```
void fasp_dbsr_alloc (
            const INT ROW,
            const INT COL,
            const INT NNZ,
            const INT nb,
            const INT storage_manner,
            dBSRmat * A )
```

Allocate memory space for BSR format sparse matrix.

Parameters

ROW	Number of rows of block
COL	Number of columns of block
NNZ	Number of nonzero blocks
nb	Dimension of each block
storage_manner	Storage manner for each sub-block
A	Pointer to new dBSRmat matrix

Author

Xiaozhe Hu

Date

10/26/2010

Definition at line 96 of file BlaSparseBSR.c.

9.27.2.2 fasp_dbsr_cp()

```
void fasp_dbsr_cp (
            const dBSRmat * A,
            dBSRmat * B )
```

copy a dCSRmat to a new one B=A

Parameters

A	Pointer to the dBSRmat matrix
B	Pointer to the dBSRmat matrix

Author

Xiaozhe Hu

Date

08/07/2011

Definition at line 168 of file BlaSparseBSR.c.

9.27.2.3 fasp_dbsr_create()

```
dBSRmat fasp_dbsr_create (
            const INT ROW,
            const INT COL,
            const INT NNZ,
            const INT nb,
            const INT storage_manner )
```

Create BSR sparse matrix data memory space.

Parameters

ROW	Number of rows of block
COL	Number of columns of block
NNZ	Number of nonzero blocks
nb	Dimension of each block
storage_manner	Storage manner for each sub-block

Returns

A The new dBSRmat matrix

Author

Xiaozhe Hu

Date

10/26/2010

Definition at line 45 of file BlaSparseBSR.c.

9.27.2.4 fasp_dbsr_diaginv()

```
dBSRmat fasp_dbsr_diaginv (
            const dBSRmat * A )
```

Compute B := D^{-1}*A, where 'D' is the block diagonal part of A.

Parameters

A	Pointer to the dBSRmat matrix

Author

Zhiyang Zhou

Date

2010/10/26

Note

Works for general nb (Xiaozhe)

Modified by Chunsheng Feng, Zheng Li on 08/25/2012

Definition at line 589 of file BlaSparseBSR.c.

9.27.2.5 fasp_dbsr_diaginv2()

```
dBSRmat fasp_dbsr_diaginv2 (
            const dBSRmat * A,
            REAL * diaginv )
```

Compute B := D^{-1}*A, where 'D' is the block diagonal part of A.

Parameters

A	Pointer to the dBSRmat matrix
diaginv	Pointer to the inverses of all the diagonal blocks

Author

Zhiyang Zhou

Date

2010/11/07

Note

Works for general nb (Xiaozhe)

Modified by Chunsheng Feng, Zheng Li on 08/25/2012

Definition at line 753 of file BlaSparseBSR.c.

9.27.2.6 fasp_dbsr_diaginv3()

```
dBSRmat fasp_dbsr_diaginv3 (
            const dBSRmat * A,
            REAL * diaginv )
```

Compute B := D^{-1}*A, where 'D' is the block diagonal part of A.

Parameters

A	Pointer to the dBSRmat matrix
diaginv	Pointer to the inverses of all the diagonal blocks

Returns

BSR matrix after diagonal scaling

Author

Xiaozhe Hu

Date

12/25/2010

Note

Works for general nb (Xiaozhe)

Modified by Xiaozhe Hu on 05/26/2012

Definition at line 855 of file BlaSparseBSR.c.

9.27.2.7 fasp_dbsr_diaginv4()

```
dBSRmat fasp_dbsr_diaginv4 (
            const dBSRmat * A,
            REAL * diaginv )
```

Compute B := D^{-1}*A, where 'D' is the block diagonal part of A.

Parameters

A	Pointer to the dBSRmat matrix
diaginv	Pointer to the inverses of all the diagonal blocks

Returns

BSR matrix after diagonal scaling

Note

Works for general nb (Xiaozhe)
A is pre-ordered that the first block of each row is the diagonal block!

Author

Xiaozhe Hu

Date

03/12/2011

Modified by Chunsheng Feng, Zheng Li on 08/26/2012

Definition at line 1213 of file BlaSparseBSR.c.

9.27.2.8 fasp_dbsr_diagLU()

```
dBSRmat fasp_dbsr_diagLU (
            const dBSRmat * A,
            REAL * DL,
            REAL * DU )
```

Compute B := DL*A*DU. We decompose each diagonal block of A into LDU form and DL = diag(L^{-1}) and DU = diag(U^{-1}).

Parameters

A	Pointer to the dBSRmat matrix
DL	Pointer to the diag(L^{-1})
DU	Pointer to the diag(U^{-1})

Returns

BSR matrix after scaling

Author

Xiaozhe Hu

Date

04/02/2014

Definition at line 1542 of file BlaSparseBSR.c.

9.27.2.9 fasp_dbsr_diagLU2()

```
dBSRmat fasp_dbsr_diagLU2 (
            dBSRmat * A,
            REAL * DL,
            REAL * DU )
```

Compute B := DL*A*DU. We decompose each diagonal block of A into LDU form and DL = diag(L^{-1}) and DU = diag(U^{-1}).

Parameters

A	Pointer to the dBSRmat matrix
DL	Pointer to the diag(L^{-1})
DU	Pointer to the diag(U^{-1})

Returns

BSR matrix after scaling

Author

Zheng Li, Xiaozhe Hu

Date

06/17/2014

Definition at line 1771 of file BlaSparseBSR.c.

9.27.2.10 fasp_dbsr_diagpref()

```
SHORT fasp_dbsr_diagpref (
            dBSRmat * A )
```

Reorder the column and data arrays of a square BSR matrix, so that the first entry in each row is the diagonal one.

Parameters

A	Pointer to the BSR matrix

Author

Xiaozhe Hu

Date

03/10/2011

Author

Chunsheng Feng, Zheng Li

Date

09/02/2012

Note

Reordering is done in place.

Definition at line 383 of file BlaSparseBSR.c.

9.27.2.11 fasp_dbsr_free()

```
void fasp_dbsr_free (
            dBSRmat * A )
```

Free memory space for BSR format sparse matrix.

Parameters

A	Pointer to the dBSRmat matrix

Author

Xiaozhe Hu

Date

10/26/2010

Definition at line 142 of file BlaSparseBSR.c.

9.27.2.12 fasp_dbsr_getblk()

```
SHORT fasp_dbsr_getblk (
            const dBSRmat * A,
            const INT * Is,
            const INT * Js,
            const INT m,
            const INT n,
            dBSRmat * B )
```

Get a sub BSR matrix of A with specified rows and columns.

Parameters

A	Pointer to dBSRmat BSR matrix
B	Pointer to dBSRmat BSR matrix
Is	Pointer to selected rows
Js	Pointer to selected columns
m	Number of selected rows
n	Number of selected columns

Returns

FASP_SUCCESS if succeeded, otherwise return error information.

Author

Shiquan Zhang, Xiaozhe Hu

Date

12/25/2010

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

Definition at line 283 of file BlaSparseBSR.c.

9.27.2.13 fasp_dbsr_getdiag()

```
void fasp_dbsr_getdiag (
            INT n,
            const dBSRmat * A,
            REAL * diag )
```

Abstract the diagonal blocks of a BSR matrix.

Parameters

n	Number of blocks to get
A	Pointer to the 'dBSRmat' type matrix
diag	Pointer to array which stores the diagonal blocks in row by row manner

Author

Zhiyang Zhou

Date

2010/10/26

Note

Works for general nb (Xiaozhe)

Modified by Chunsheng Feng, Zheng Li on 08/25/2012

Definition at line 1504 of file BlaSparseBSR.c.

9.27.2.14 fasp_dbsr_getdiaginv()

```
dvector fasp_dbsr_getdiaginv (
            const dBSRmat * A )
```

Get D^{-1} of matrix A.

Parameters

A	Pointer to the dBSRmat matrix

Author

Xiaozhe Hu

Date

02/19/2013

Note

Works for general nb (Xiaozhe)

Definition at line 483 of file BlaSparseBSR.c.

9.27.2.15 fasp_dbsr_perm()

```
dBSRmat fasp_dbsr_perm (
            const dBSRmat * A,
            const INT * P )
```

Apply permutation of A, i.e. Aperm=PAP' by the orders given in P.

Parameters

A	Pointer to the original dCSRmat matrix
P	Pointer to the given ordering

Returns

The new ordered dCSRmat matrix if succeed, NULL if fail

Author

Zheng Li

Date

24/9/2015

Note

P[i] = k means k-th row and column become i-th row and column!

Definition at line 1972 of file BlaSparseBSR.c.

9.27.2.16 fasp_dbsr_trans()

```
INT fasp_dbsr_trans (
            const 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 195 of file BlaSparseBSR.c.

9.28 BlaSparseCheck.c File Reference

Check properties of sparse matrices.

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

Functions

- INT fasp_check_diagpos (const dCSRmat *A)

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

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

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

    Check symmetry of a sparse matrix of CSR format.
- void fasp_check_dCSRmat (const dCSRmat *A)

    Check whether an dCSRmat matrix is supported or not.
- SHORT fasp_check_iCSRmat (const iCSRmat *A)

    Check whether an iCSRmat matrix is valid or not.

9.28.1 Detailed Description

Check properties of sparse matrices.

Note

This file contains Level-1 (Bla) functions. It requires: AuxMemory.c, AuxMessage.c, AuxVector.c, and BlaSparse←CSR.c

9.28.2 Function Documentation

9.28.2.1 fasp_check_dCSRmat()

```
void fasp_check_dCSRmat (
            const dCSRmat * A )
```

Check whether an dCSRmat matrix is supported or not.

Parameters

A	Pointer to the matrix in dCSRmat format

Author

Chensong Zhang

Date

03/29/2009

Definition at line 283 of file BlaSparseCheck.c.

9.28.2.2 fasp_check_diagdom()

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

Check whether a matrix is diagonal dominant.

INT fasp_check_diagdom (const dCSRmat ∗A)

Parameters

A	Pointer to the dCSRmat matrix

Returns

Number of the rows which are diagonal dominant

Note

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

Author

Shuo Zhang

Date

03/29/2009

Definition at line 116 of file BlaSparseCheck.c.

9.28.2.3 fasp_check_diagpos()

```
INT fasp_check_diagpos (
            const 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 35 of file BlaSparseCheck.c.

9.28.2.4 fasp_check_diagzero()

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

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

Parameters

A	pointr to the dCSRmat matrix

Returns

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

Author

Shuo Zhang

Date

03/29/2009

Definition at line 72 of file BlaSparseCheck.c.

9.28.2.5 fasp_check_iCSRmat()

```
SHORT fasp_check_iCSRmat (
            const 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 320 of file BlaSparseCheck.c.

9.28.2.6 fasp_check_symm()

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

Check symmetry of a sparse matrix of CSR format.

Parameters

A	Pointer to the dCSRmat matrix

Returns

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

Note

Print the maximal relative difference between matrix and its transpose.

Author

Shuo Zhang

Date

03/29/2009

Definition at line 161 of file BlaSparseCheck.c.

9.29 BlaSparseCOO.c File Reference

Sparse matrix operations for dCOOmat matrices.

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

Functions

- dCOOmat fasp_dcoo_create (const INT m, const INT n, const INT nnz)

    Create IJ sparse matrix data memory space.
- void fasp_dcoo_alloc (const INT m, const INT n, const INT nnz, dCOOmat *A)

    Allocate COO sparse matrix memory space.
- void fasp_dcoo_free (dCOOmat *A)

    Free IJ sparse matrix data memory space.
- void fasp_dcoo_shift (dCOOmat *A, const INT offset)

    Re-index a REAL matrix in IJ format to make the index starting from 0 or 1.

9.29.1 Detailed Description

Sparse matrix operations for dCOOmat matrices.

Note

This file contains Level-1 (Bla) functions. It requires: AuxMemory.c and AuxThreads.c

9.29.2 Function Documentation

9.29.2.1 fasp_dcoo_alloc()

```
void fasp_dcoo_alloc (
            const INT m,
            const INT n,
            const INT nnz,
            dCOOmat * A )
```

Allocate COO sparse matrix memory space.

Parameters

m	Number of rows
n	Number of columns
nnz	Number of nonzeros
A	Pointer to the dCSRmat matrix

Author

Xiaozhe Hu

Date

03/25/2013

Definition at line 70 of file BlaSparseCOO.c.

9.29.2.2 fasp_dcoo_create()

```
dCOOmat fasp_dcoo_create (
            const INT m,
            const INT n,
            const INT nnz )
```

Create IJ sparse matrix data memory space.

Parameters

m	Number of rows
n	Number of columns
nnz	Number of nonzeros

Returns

A The new dCOOmat matrix

Author

Chensong Zhang

Date

2010/04/06

Definition at line 42 of file BlaSparseCOO.c.

9.29.2.3 fasp_dcoo_free()

```
void fasp_dcoo_free (
            dCOOmat * A )
```

Free IJ sparse matrix data memory space.

Parameters

A	Pointer to the dCOOmat matrix

Author

Chensong Zhang

Date

2010/04/03

Definition at line 102 of file BlaSparseCOO.c.

9.29.2.4 fasp_dcoo_shift()

```
void fasp_dcoo_shift (
            dCOOmat * A,
            const INT offset )
```

Re-index a REAL matrix in IJ format to make the index starting from 0 or 1.

Parameters

A	Pointer to IJ matrix
offset	Size of offset (1 or -1)

Author

Chensong Zhang

Date

2010/04/06

Modified by Chunsheng Feng, Zheng Li on 08/25/2012

Definition at line 124 of file BlaSparseCOO.c.

9.30 BlaSparseCSR.c File Reference

Sparse matrix operations for dCSRmat matrices.

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

Functions

- dCSRmat fasp_dcsr_create (const INT m, const INT n, const INT nnz)

  Create CSR sparse matrix data memory space.
- iCSRmat fasp_icsr_create (const INT m, const INT n, const INT nnz)

  Create CSR sparse matrix data memory space.
- void fasp_dcsr_alloc (const INT m, const INT n, const INT nnz, dCSRmat ∗A)

  Allocate CSR sparse matrix memory space.
- void fasp_dcsr_free (dCSRmat ∗A)

  Free CSR sparse matrix data memory space.
- void fasp_icsr_free (iCSRmat ∗A)

  Free CSR sparse matrix data memory space.
- INT fasp_dcsr_bandwidth (const dCSRmat ∗A)

  Get bandwith of 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.
- SHORT fasp_dcsr_getblk (const dCSRmat ∗A, const INT ∗Is, const INT ∗Js, const INT m, const INT n, dCSRmat ∗B)

  Get a sub CSR matrix of A with specified rows and columns.
- void fasp_dcsr_getdiag (INT n, const dCSRmat ∗A, dvector ∗diag)

  Get first n diagonal entries of a CSR matrix A.
- void fasp_dcsr_getcol (const INT n, const 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, const REAL value)

  Regularize diagonal entries of a CSR sparse matrix.
- void fasp_icsr_cp (const iCSRmat ∗A, iCSRmat ∗B)

  Copy a iCSRmat to a new one B=A.
- void fasp_dcsr_cp (const dCSRmat ∗A, dCSRmat ∗B)

  copy a dCSRmat to a new one B=A
- void fasp_icsr_trans (const iCSRmat ∗A, iCSRmat ∗AT)

  Find transpose of iCSRmat matrix A.
- INT fasp_dcsr_trans (const dCSRmat ∗A, dCSRmat ∗AT)

  Find transpose of dCSRmat matrix A.

- void **fasp_dcsr_transpose** (INT *row[2], INT *col[2], REAL *val[2], INT *nn, INT *tniz)
- void fasp_dcsr_compress (const dCSRmat *A, dCSRmat *B, const REAL dtol)

    Compress a CSR matrix A and store in CSR matrix B by dropping small entries abs(aij)<=dtol.
- SHORT fasp_dcsr_compress_inplace (dCSRmat *A, const REAL dtol)

    Compress a CSR matrix A IN PLACE by dropping small entries abs(aij)<=dtol.
- void fasp_dcsr_shift (dCSRmat *A, const INT offset)

    Re-index a REAL matrix in CSR format to make the index starting from 0 or 1.
- void fasp_dcsr_symdiagscale (dCSRmat *A, const dvector *diag)

    Symmetric diagonal scaling D^{-1/2}AD^{-1/2}.
- dCSRmat fasp_dcsr_sympart (dCSRmat *A)

    Get symmetric part of a dCSRmat matrix.
- void fasp_dcsr_multicoloring (dCSRmat *A, INT *flags, INT *groups)

    Use the greedy multi-coloring to get color groups of the adjacency graph of A.
- void fasp_dcsr_transz (dCSRmat *A, INT *p, dCSRmat *AT)

    Generalized transpose of A: (n x m) matrix given in dCSRmat format.
- dCSRmat fasp_dcsr_permz (dCSRmat *A, INT *p)

    Permute rows and cols of A, i.e. A=PAP' by the ordering in p.
- void fasp_dcsr_sortz (dCSRmat *A, const SHORT isym)

    Sort each row of A in ascending order w.r.t. column indices.

9.30.1 Detailed Description

Sparse matrix operations for dCSRmat matrices.

Note

This file contains Level-1 (Bla) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, AuxSort.c, AuxThreads.c, AuxVector.c, and BlaSpmvCSR.c

9.30.2 Function Documentation

9.30.2.1 fasp_dcsr_alloc()

```
void fasp_dcsr_alloc (
            const INT m,
            const INT n,
            const INT nnz,
            dCSRmat * A )
```

Allocate CSR sparse matrix memory space.

Parameters

m	Number of rows
n	Number of columns
nnz	Number of nonzeros
A	Pointer to the dCSRmat matrix

Author

Chensong Zhang

Date

2010/04/06

Definition at line 134 of file BlaSparseCSR.c.

9.30.2.2 fasp_dcsr_bandwidth()

```
INT fasp_dcsr_bandwidth (
            const dCSRmat * A )
```

Get bandwith of matrix.

Parameters

A	pointer to the dCSRmat matrix

Author

Zheng Li

Date

03/22/2015

Definition at line 213 of file BlaSparseCSR.c.

9.30.2.3 fasp_dcsr_compress()

```
void fasp_dcsr_compress (
            const dCSRmat * A,
            dCSRmat * B,
            const REAL dtol )
```

Compress a CSR matrix A and store in CSR matrix B by dropping small entries abs(aij)<=dtol.

Parameters

A	Pointer to dCSRmat CSR matrix
B	Pointer to dCSRmat CSR matrix
dtol	Drop tolerance

Author

Shiquan Zhang

Date

03/10/2010

Modified by Chunsheng Feng, Zheng Li on 08/25/2012

Definition at line 1043 of file BlaSparseCSR.c.

9.30.2.4 fasp_dcsr_compress_inplace()

```
SHORT fasp_dcsr_compress_inplace (
            dCSRmat * A,
            const REAL dtol )
```

Compress a CSR matrix A IN PLACE by dropping small entries abs(aij)<=dtol.

Parameters

A	Pointer to dCSRmat CSR matrix
dtol	Drop tolerance

Author

Xiaozhe Hu

Date

12/25/2010

Modified by Chensong Zhang on 02/21/2013

Note

This routine can be modified for filtering.

Definition at line 1123 of file BlaSparseCSR.c.

9.30.2.5 fasp_dcsr_cp()

```
void fasp_dcsr_cp (
            const dCSRmat * A,
            dCSRmat * B )
```

copy a dCSRmat to a new one B=A

Parameters

A	Pointer to the dCSRmat matrix
B	Pointer to the dCSRmat matrix

Author

Chensong Zhang

Date

04/06/2010

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

Definition at line 811 of file BlaSparseCSR.c.

9.30.2.6 fasp_dcsr_create()

```
dCSRmat fasp_dcsr_create (
            const INT m,
            const INT n,
            const INT nnz )
```

Create CSR sparse matrix data memory space.

Parameters

m	Number of rows
n	Number of columns
nnz	Number of nonzeros

Returns

A the new dCSRmat matrix

Author

Chensong Zhang

Date

2010/04/06

Definition at line 43 of file BlaSparseCSR.c.

9.30.2.7 fasp_dcsr_diagpref()

```
void fasp_dcsr_diagpref (
            dCSRmat * A )
```

Re-order the column and data arrays of a CSR matrix, so that the first entry in each row is the diagonal.

Parameters

A	Pointer to the matrix to be re-ordered

Author

Zhiyang Zhou

Date

09/09/2010

Author

Chunsheng Feng, Zheng Li

Date

09/02/2012

Note

Reordering is done in place.

Modified by Chensong Zhang on Dec/21/2012

Definition at line 641 of file BlaSparseCSR.c.

9.30.2.8 fasp_dcsr_free()

```
void fasp_dcsr_free (
            dCSRmat * A )
```

Free CSR sparse matrix data memory space.

Parameters

A	Pointer to the dCSRmat matrix

Author

Chensong Zhang

Date

2010/04/06

Definition at line 175 of file BlaSparseCSR.c.

9.30.2.9 fasp_dcsr_getblk()

```
SHORT fasp_dcsr_getblk (
            const dCSRmat * A,
            const INT * Is,
            const INT * Js,
            const INT m,
            const INT n,
            dCSRmat * B )
```

Get a sub CSR matrix of A with specified rows and columns.

Parameters

A	Pointer to dCSRmat matrix
B	Pointer to dCSRmat matrix
Is	Pointer to selected rows
Js	Pointer to selected columns
m	Number of selected rows
n	Number of selected columns

Returns

FASP_SUCCESS if succeeded, otherwise return error information.

Author

Shiquan Zhang, Xiaozhe Hu

Date

12/25/2010

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

Definition at line 412 of file BlaSparseCSR.c.

9.30.2.10 fasp_dcsr_getcol()

```
void fasp_dcsr_getcol (
            const INT n,
            const dCSRmat * A,
            REAL * col )
```

Get the n-th column of a CSR matrix A.

Parameters

n	Index of a column of A (0 <= n <= A.col-1)
A	Pointer to dCSRmat CSR matrix
col	Pointer to the column

Author

Xiaozhe Hu

Date

11/07/2009

Modified by Chunsheng Feng, Zheng Li on 07/08/2012

Definition at line 562 of file BlaSparseCSR.c.

9.30.2.11 fasp_dcsr_getdiag()

```
void fasp_dcsr_getdiag (
            INT n,
            const dCSRmat * A,
            dvector * diag )
```

Get first n diagonal entries of a CSR matrix A.

Parameters

n	Number of diagonal entries to get (if n=0, then get all diagonal entries)
A	Pointer to dCSRmat CSR matrix
diag	Pointer to the diagonal as a dvector

Author

Chensong Zhang

Date

05/20/2009

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

Definition at line 498 of file BlaSparseCSR.c.

9.30.2.12 fasp_dcsr_multicoloring()

```
void fasp_dcsr_multicoloring (
            dCSRmat * A,
            INT * flags,
            INT * groups )
```

Use the greedy multi-coloring to get color groups of the adjacency graph of A.

Parameters

A	Input dCSRmat
flags	flags for the independent group
groups	Return group numbers

Author

Chunsheng Feng

Date

09/15/2012

Definition at line 1351 of file BlaSparseCSR.c.

9.30.2.13 fasp_dcsr_perm()

```
dCSRmat fasp_dcsr_perm (
            dCSRmat * A,
            INT * P )
```

Apply permutation of A, i.e. Aperm=PAP' by the orders given in P.

Parameters

A	Pointer to the original dCSRmat matrix
P	Pointer to orders

Returns

The new ordered dCSRmat matrix if succeed, NULL if fail

Author

Shiquan Zhang

Date

03/10/2010

Note

P[i] = k means k-th row and column become i-th row and column!
Deprecated! Will be replaced by fasp_dcsr_permz later. –Chensong

Modified by Chunsheng Feng, Zheng Li on 07/12/2012

Definition at line 243 of file BlaSparseCSR.c.

9.30.2.14 fasp_dcsr_permz()

```
dCSRmat fasp_dcsr_permz (
            dCSRmat * A,
            INT * p )
```

Permute rows and cols of A, i.e. A=PAP' by the ordering in p.

Parameters

A	Pointer to the original dCSRmat matrix
p	Pointer to ordering

Note

This is just applying twice fasp_dcsr_transz(&A,p,At).
In matlab notation: Aperm=A(p,p);

Returns

The new ordered dCSRmat matrix if succeed, NULL if fail

Author

Ludmil Zikatanov

Date

19951219 (Fortran), 20150912 (C)

Definition at line 1572 of file BlaSparseCSR.c.

9.30.2.15 fasp_dcsr_regdiag()

```
SHORT fasp_dcsr_regdiag (
            dCSRmat * A,
            const REAL value )
```

Regularize diagonal entries of a CSR sparse matrix.

Parameters

A	Pointer to the dCSRmat matrix
value	Set a value on diag(A) which is too close to zero to "value"

Returns

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

Author

Shiquan Zhang

Date

11/07/2009

Definition at line 747 of file BlaSparseCSR.c.

9.30.2.16 fasp_dcsr_shift()

```
void fasp_dcsr_shift (
            dCSRmat * A,
            const INT offset )
```

Re-index a REAL matrix in CSR format to make the index starting from 0 or 1.

Parameters

A	Pointer to CSR matrix
offset	Size of offset (1 or -1)

Author

Chensong Zhang

Date

04/06/2010

Modified by Chunsheng Feng, Zheng Li on 07/11/2012

Definition at line 1171 of file BlaSparseCSR.c.

9.30.2.17 fasp_dcsr_sort()

```
void fasp_dcsr_sort (
            dCSRmat * A )
```

Sort each row of A in ascending order w.r.t. column indices.

Parameters

A	Pointer to the dCSRmat matrix

Author

Shiquan Zhang

Date

06/10/2010

Definition at line 354 of file BlaSparseCSR.c.

9.30.2.18 fasp_dcsr_sortz()

```
void fasp_dcsr_sortz (
            dCSRmat * A,
            const SHORT isym )
```

Sort each row of A in ascending order w.r.t. column indices.

Parameters

A	Pointer to the dCSRmat matrix
isym	Flag for symmetry, =[0/nonzero]=[general/symmetric] matrix

Note

Applying twice fasp_dcsr_transz(), if A is symmetric, then the transpose is applied only once and then AT copied on A.

Author

Ludmil Zikatanov

Date

19951219 (Fortran), 20150912 (C)

Definition at line 1604 of file BlaSparseCSR.c.

9.30.2.19 fasp_dcsr_symdiagscale()

```
void fasp_dcsr_symdiagscale (
            dCSRmat * A,
            const dvector * diag )
```

Symmetric diagonal scaling D^{-1/2}AD^{-1/2}.

Parameters

A	Pointer to the dCSRmat matrix
diag	Pointer to the diagonal entries

Author

Xiaozhe Hu

Date

01/31/2011

Modified by Chunsheng Feng, Zheng Li on 07/11/2012

Definition at line 1232 of file BlaSparseCSR.c.

9.30.2.20 fasp_dcsr_sympart()

```
dCSRmat fasp_dcsr_sympart (
            dCSRmat * A )
```

Get symmetric part of a dCSRmat matrix.

Parameters

A	Pointer to the dCSRmat matrix

Returns

Symmetrized the dCSRmat matrix

Author

Xiaozhe Hu

Date

03/21/2011

Definition at line 1318 of file BlaSparseCSR.c.

9.30.2.21 fasp_dcsr_trans()

```
void fasp_dcsr_trans (
            const dCSRmat * A,
            dCSRmat * AT )
```

Find transpose of dCSRmat matrix A.

Parameters

A	Pointer to the dCSRmat matrix
AT	Pointer to the transpose of dCSRmat matrix A (output)

Author

Chensong Zhang

Date

04/06/2010

Modified by Chunsheng Feng, Zheng Li on 06/20/2012

Definition at line 912 of file BlaSparseCSR.c.

9.30.2.22 fasp_dcsr_transz()

```
void fasp_dcsr_transz (
            dCSRmat * A,
            INT * p,
            dCSRmat * AT )
```

Generalized transpose of A: (n x m) matrix given in dCSRmat format.

Parameters

A	Pointer to matrix in dCSRmat for transpose, INPUT
p	Permutation, INPUT
AT	Pointer to matrix AT = transpose(A) if p = NULL, OR AT = transpose(A)p if p is not NULL

Note

The storage for all pointers in AT should already be allocated, i.e. AT->IA, AT->JA and AT->val should be allocated before calling this function. If A.val=NULL, then AT->val[] is not changed.

performs AT=transpose(A)p, where p is a permutation. If p=NULL then p=I is assumed. Applying twice this procedure one gets At=transpose(transpose(A)p)p = transpose(p)Ap, which is the same A with rows and columns permutted according to p.

If A=NULL, then only transposes/permutes the structure of A.

For p=NULL, applying this two times A–>AT–>A orders all the row indices in A in increasing order.

Reference: Fred G. Gustavson. Two fast algorithms for sparse matrices: multiplication and permuted transposition. ACM Trans. Math. Software, 4(3):250–269, 1978.

Author

Ludmil Zikatanov

Date

19951219 (Fortran), 20150912 (C)

Definition at line 1452 of file BlaSparseCSR.c.

9.30.2.23 fasp_icsr_cp()

```
void fasp_icsr_cp (
            const iCSRmat * A,
            iCSRmat * B )
```

Copy a iCSRmat to a new one B=A.

Parameters

A	Pointer to the iCSRmat matrix
B	Pointer to the iCSRmat matrix

Author

Chensong Zhang

Date

05/16/2013

Definition at line 786 of file BlaSparseCSR.c.

9.30.2.24 fasp_icsr_create()

```
iCSRmat fasp_icsr_create (
            const INT m,
            const INT n,
            const INT nnz )
```

Create CSR sparse matrix data memory space.

Parameters

m	Number of rows
n	Number of columns
nnz	Number of nonzeros

Returns

A the new iCSRmat matrix

Author

Chensong Zhang

Date

2010/04/06

Definition at line 89 of file BlaSparseCSR.c.

9.30.2.25 fasp_icsr_free()

```
void fasp_icsr_free (
            iCSRmat * A )
```

Free CSR sparse matrix data memory space.

Parameters

A	Pointer to the iCSRmat matrix

Author

Chensong Zhang

Date

2010/04/06

Definition at line 194 of file BlaSparseCSR.c.

9.30.2.26 fasp_icsr_trans()

```
void fasp_icsr_trans (
            const iCSRmat * A,
            iCSRmat * AT )
```

Find transpose of iCSRmat matrix A.

Parameters

A	Pointer to the iCSRmat matrix A
AT	Pointer to the iCSRmat matrix A'

Author

Chensong Zhang

Date

04/06/2010

Modified by Chunsheng Feng, Zheng Li on 06/20/2012

Definition at line 836 of file BlaSparseCSR.c.

9.31 BlaSparseCSRL.c File Reference

Sparse matrix operations for dCSRLmat matrices.

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

Functions

- dCSRLmat * fasp_dcsrl_create (const INT num_rows, const INT num_cols, const INT num_nonzeros)

    Create a dCSRLmat object.
- void fasp_dcsrl_free (dCSRLmat *A)

    Destroy a dCSRLmat object.

9.31.1 Detailed Description

Sparse matrix operations for dCSRLmat matrices.

Note

This file contains Level-1 (Bla) functions. It requires: AuxMemory.c

Reference: John Mellor-Crummey and John Garvin Optimizaing sparse matrix vector product computations using unroll and jam, Tech Report Rice Univ, Aug 2002.

9.31.2 Function Documentation

9.31.2.1 fasp_dcsrl_create()

```
dCSRLmat * fasp_dcsrl_create (
            const INT num_rows,
            const INT num_cols,
            const INT num_nonzeros )
```

Create a dCSRLmat object.

Parameters

num_rows	Number of rows
num_cols	Number of cols
num_nonzeros	Number of nonzero entries

Author

Zhiyang Zhou

Date

01/07/2011

Definition at line 39 of file BlaSparseCSRL.c.

9.31.2.2 fasp_dcsrl_free()

```
void fasp_dcsrl_free (
            dCSRLmat * A )
```

Destroy a dCSRLmat object.

Parameters

A	Pointer to the dCSRLmat type matrix

Author

Zhiyang Zhou

Date

01/07/2011

Definition at line 67 of file BlaSparseCSRL.c.

9.32 BlaSparseSTR.c File Reference

Sparse matrix operations for dSTRmat matrices.

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

Functions

- dSTRmat fasp_dstr_create (const INT nx, const INT ny, const INT nz, const INT nc, const INT nband, INT *offsets)

    Create STR sparse matrix data memory space.
- void fasp_dstr_alloc (const INT nx, const INT ny, const INT nz, const INT nxy, const INT ngrid, const INT nband, const INT nc, INT *offsets, dSTRmat *A)

    Allocate STR sparse matrix memory space.
- void fasp_dstr_free (dSTRmat *A)

    Free STR sparse matrix data memeory space.
- void fasp_dstr_cp (const dSTRmat *A, dSTRmat *B)

    Copy a dSTRmat to a new one B=A.

9.32.1 Detailed Description

Sparse matrix operations for dSTRmat matrices.

Note

This file contains Level-1 (Bla) functions. It requires: AuxMemory.c

9.32.2 Function Documentation

9.32.2.1 fasp_dstr_alloc()

```
void fasp_dstr_alloc (
            const INT nx,
            const INT ny,
            const INT nz,
            const INT nxy,
            const INT ngrid,
            const INT nband,
            const INT nc,
            INT * offsets,
            dSTRmat * A )
```

Allocate STR sparse matrix memory space.

Parameters

nx	Number of grids in x direction
ny	Number of grids in y direction
nz	Number of grids in z direction
nxy	Number of grids in x-y plane
ngrid	Number of grids
nband	Number of off-diagonal bands
nc	Number of components
offsets	Shift from diagonal
A	Pointer to the dSTRmat matrix

Author

Shiquan Zhang, Xiaozhe Hu

Date

05/17/2010

Definition at line 93 of file BlaSparseSTR.c.

9.32.2.2 fasp_dstr_cp()

```
void fasp_dstr_cp (
            const dSTRmat * A,
            dSTRmat * B )
```

Copy a dSTRmat to a new one B=A.

Parameters

A	Pointer to the dSTRmat matrix
B	Pointer to the dSTRmat matrix

Author

Zhiyang Zhou

Date

04/21/2010

Definition at line 165 of file BlaSparseSTR.c.

9.32.2.3 fasp_dstr_create()

```
dSTRmat fasp_dstr_create (
            const INT nx,
            const INT ny,
            const INT nz,
            const INT nc,
            const INT nband,
            INT * offsets )
```

Create STR sparse matrix data memory space.

Parameters

nx	Number of grids in x direction
ny	Number of grids in y direction
nz	Number of grids in z direction
nc	Number of components
nband	Number of off-diagonal bands
offsets	Shift from diagonal

Returns

The dSTRmat matrix

Author

Shiquan Zhang, Xiaozhe Hu

Date

05/17/2010

Definition at line 41 of file BlaSparseSTR.c.

9.32.2.4 fasp_dstr_free()

```
void fasp_dstr_free (
            dSTRmat * A )
```

Free STR sparse matrix data memeory space.

Parameters

A	Pointer to the dSTRmat matrix

Author

Shiquan Zhang, Xiaozhe Hu

Date

05/17/2010

Definition at line 136 of file BlaSparseSTR.c.

9.33 BlaSparseUtil.c File Reference

Routines for sparse matrix operations.

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

Functions

- void fasp_sparse_abybms_ (INT ∗ia, INT ∗ja, INT ∗ib, INT ∗jb, INT ∗nap, INT ∗map, INT ∗mbp, INT ∗ic, INT ∗jc)

  Multiplication of two sparse matrices: calculating the nonzero structure of the result if jc is not null. If jc is null only finds num of nonzeroes.
- void fasp_sparse_abyb_ (INT ∗ia, INT ∗ja, REAL ∗a, INT ∗ib, INT ∗jb, REAL ∗b, INT ∗nap, INT ∗map, INT ∗mbp, INT ∗ic, INT ∗jc, REAL ∗c)

  Multiplication of two sparse matrices: calculating the numerical values in the result.
- void fasp_sparse_iit_ (INT ∗ia, INT ∗ja, INT ∗na, INT ∗ma, INT ∗iat, INT ∗jat)

  Transpose a boolean matrix (only given by ia, ja)
- void fasp_sparse_aat_ (INT ∗ia, INT ∗ja, REAL ∗a, INT ∗na, INT ∗ma, INT ∗iat, INT ∗jat, REAL ∗at)

  transpose a boolean matrix (only given by ia, ja)
- void fasp_sparse_aplbms_ (INT ∗ia, INT ∗ja, INT ∗ib, INT ∗jb, INT ∗nab, INT ∗mab, INT ∗ic, INT ∗jc)

  Addition of two sparse matrices: calculating the nonzero structure of the result if jc is not null. if jc is null only finds num of nonzeroes.
- void fasp_sparse_aplusb_ (INT ∗ia, INT ∗ja, REAL ∗a, INT ∗ib, INT ∗jb, REAL ∗b, INT ∗nab, INT ∗mab, INT ∗ic, INT ∗jc, REAL ∗c)

  Addition of two sparse matrices: calculating the numerical values in the result.
- void fasp_sparse_rapms_ (INT ∗ir, INT ∗jr, INT ∗ia, INT ∗ja, INT ∗ip, INT ∗jp, INT ∗nin, INT ∗ncin, INT ∗iac, INT ∗jac, INT ∗maxrout)

  Calculates the nonzero structure of R∗A∗P, if jac is not null. If jac is null only finds num of nonzeroes.
- void fasp_sparse_wtams_ (INT ∗jw, INT ∗ia, INT ∗ja, INT ∗nwp, INT ∗map, INT ∗jv, INT ∗nvp, INT ∗icp)

  Finds the nonzeroes in the result of v^t = w^t A, where w is a sparse vector and A is sparse matrix. jv is an integer array containing the indices of the nonzero elements in the result.
- void fasp_sparse_wta_ (INT ∗jw, REAL ∗w, INT ∗ia, INT ∗ja, REAL ∗a, INT ∗nwp, INT ∗map, INT ∗jv, REAL ∗v, INT ∗nvp)

  Calculate v^t = w^t A, where w is a sparse vector and A is sparse matrix. v is an array of dimension = number of columns in A.
- void fasp_sparse_ytxbig_ (INT ∗jy, REAL ∗y, INT ∗nyp, REAL ∗x, REAL ∗s)

  Calculates s = y^t x. y-sparse, x - no.
- void fasp_sparse_ytx_ (INT ∗jy, REAL ∗y, INT ∗jx, REAL ∗x, INT ∗nyp, INT ∗nxp, INT ∗icp, REAL ∗s)

  Calculates s = y^t x. y is sparse, x is sparse.
- void fasp_sparse_rapcmp_ (INT ∗ir, INT ∗jr, REAL ∗r, INT ∗ia, INT ∗ja, REAL ∗a, INT ∗ipt, INT ∗jpt, REAL ∗pt, INT ∗nin, INT ∗ncin, INT ∗iac, INT ∗jac, REAL ∗ac, INT ∗idummy)

  Calculates R∗A∗P after the nonzero structure of the result is known. iac,jac,ac have to be allocated before call to this function.
- ivector fasp_sparse_MIS (dCSRmat ∗A)

  get the maximal independet set of a CSR matrix

9.33.1 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
This file contains Level-1 (Bla) functions. It requires: AuxMemory.c

9.33.2 Function Documentation

9.33.2.1 fasp_sparse_aat_()

```
void fasp_sparse_aat_ (
            INT * ia,
            INT * ja,
            REAL * a,
            INT * na,
            INT * ma,
            INT * iat,
            INT * jat,
            REAL * at )
```

transpose a boolean matrix (only given by ia, ja)

Parameters

ia	array of row pointers (as usual in CSR)
ja	array of column indices
a	array of entries of teh input
na	number of rows of A
ma	number of cols of A
iat	array of row pointers in the result
jat	array of column indices
at	array of entries of the result

Definition at line 273 of file BlaSparseUtil.c.

9.33.2.2 fasp_sparse_abyb_()

```
void fasp_sparse_abyb_ (
            INT * ia,
            INT * ja,
            REAL * a,
            INT * ib,
            INT * jb,
            REAL * b,
            INT * nap,
            INT * map,
            INT * mbp,
            INT * ic,
            INT * jc,
            REAL * c )
```

Multiplication of two sparse matrices: calculating the numerical values in the result.

Parameters

ia	array of row pointers 1st multiplicand
ja	array of column indices 1st multiplicand
a	entries of the 1st multiplicand
ib	array of row pointers 2nd multiplicand
jb	array of column indices 2nd multiplicand
b	entries of the 2nd multiplicand
ic	array of row pointers in c=a*b
jc	array of column indices in c=a*b
c	entries of the result: c= a*b
nap	number of rows in the 1st multiplicand
map	number of columns in the 1st multiplicand
mbp	number of columns in the 2nd multiplicand

Modified by Chensong Zhang on 09/11/2012

Definition at line 125 of file BlaSparseUtil.c.

9.33.2.3 fasp_sparse_abybms_()

```
void fasp_sparse_abybms_ (
            INT * ia,
            INT * ja,
            INT * ib,
            INT * jb,
            INT * nap,
            INT * map,
```

```
INT * mbp,
INT * ic,
INT * jc )
```

Multiplication of two sparse matrices: calculating the nonzero structure of the result if jc is not null. If jc is null only finds num of nonzeroes.

Parameters

ia	array of row pointers 1st multiplicand
ja	array of column indices 1st multiplicand
ib	array of row pointers 2nd multiplicand
jb	array of column indices 2nd multiplicand
nap	number of rows of A
map	number of cols of A
mbp	number of cols of b
ic	array of row pointers in the result (this is also computed here again, so that we can have a stand alone call of this routine, if for some reason the number of nonzeros in the result is known)
jc	array of column indices in the result c=a*b

Modified by Chensong Zhang on 09/11/2012

Definition at line 54 of file BlaSparseUtil.c.

9.33.2.4 fasp_sparse_aplbms_()

```
void void fasp_sparse_aplbms_ (
INT * ia,
INT * ja,
INT * ib,
INT * jb,
INT * nab,
INT * mab,
INT * ic,
INT * jc )
```

Addition of two sparse matrices: calculating the nonzero structure of the result if jc is not null. if jc is null only finds num of nonzeroes.

Parameters

ia	array of row pointers 1st summand
ja	array of column indices 1st summand
ib	array of row pointers 2nd summand
jb	array of column indices 2nd summand
nab	number of rows
mab	number of cols
ic	array of row pointers in the result (this is also computed here again, so that we can have a stand alone call of this routine, if for some reason the number of nonzeros in the result is known)
jc	array of column indices in the result c=a+b

Definition at line 359 of file BlaSparseUtil.c.

9.33.2.5 fasp_sparse_aplusb_()

```
void fasp_sparse_aplusb_ (
            INT * ia,
            INT * ja,
            REAL * a,
            INT * ib,
            INT * jb,
            REAL * b,
            INT * nab,
            INT * mab,
            INT * ic,
            INT * jc,
            REAL * c )
```

Addition of two sparse matrices: calculating the numerical values in the result.

Parameters

ia	array of row pointers 1st summand
ja	array of column indices 1st summand
a	entries of the 1st summand
ib	array of row pointers 2nd summand
jb	array of column indices 2nd summand
b	entries of the 2nd summand
nab	number of rows
mab	number of cols
ic	array of row pointers in c=a+b
jc	array of column indices in c=a+b
c	entries of the result: c=a+b

Definition at line 431 of file BlaSparseUtil.c.

9.33.2.6 fasp_sparse_iit_()

```
void fasp_sparse_iit_ (
            INT * ia,
            INT * ja,
            INT * na,
            INT * ma,
            INT * iat,
            INT * jat )
```

Transpose a boolean matrix (only given by ia, ja)

Parameters

ia	array of row pointers (as usual in CSR)
ja	array of column indices
na	number of rows
ma	number of cols
iat	array of row pointers in the result
jat	array of column indices

Note

For the concrete algorithm, see:

Definition at line 198 of file BlaSparseUtil.c.

9.33.2.7 fasp_sparse_MIS()

```
ivector fasp_sparse_MIS (
            dCSRmat * A )
```

get the maximal independet set of a CSR matrix

Parameters

A	pointer to the matrix

Note

: only use the sparsity of A, index starts from 1 (fortran)!!

Definition at line 908 of file BlaSparseUtil.c.

9.33.2.8 fasp_sparse_rapcmp_()

```
void fasp_sparse_rapcmp_ (
            INT * ir,
            INT * jr,
            REAL * r,
            INT * ia,
            INT * ja,
            REAL * a,
```

```
INT * ipt,
INT * jpt,
REAL * pt,
INT * nin,
INT * ncin,
INT * iac,
INT * jac,
REAL * ac,
INT * idummy )
```

Calculates R∗A∗P after the nonzero structure of the result is known. iac,jac,ac have to be allocated before call to this function.

Note

:I: is input :O: is output :IO: is both

Parameters

ir	:I: array of row pointers for R
jr	:I: array of column indices for R
r	:I: entries of R
ia	:I: array of row pointers for A
ja	:I: array of column indices for A
a	:I: entries of A
ipt	:I: array of row pointers for P
jpt	:I: array of column indices for P
pt	:I: entries of P
nin	:I: number of rows in R
ncin	:I: number of rows in
iac	:O: array of row pointers for P
jac	:O: array of column indices for P
ac	:O: entries of P
idummy	not changed

Note

compute R∗A∗P for known nonzero structure of the result the result is stored in iac,jac,ac!

Definition at line 788 of file BlaSparseUtil.c.

9.33.2.9 fasp_sparse_rapms_()

```
void fasp_sparse_rapms_ (
            INT * ir,
```

```
            INT * jr,
            INT * ia,
            INT * ja,
            INT * ip,
            INT * jp,
            INT * nin,
            INT * ncin,
            INT * iac,
            INT * jac,
            INT * maxrout )
```

Calculates the nonzero structure of R∗A∗P, if jac is not null. If jac is null only finds num of nonzeroes.

Note

:I: is input :O: is output :IO: is both

Parameters

ir	:I: array of row pointers for R
jr	:I: array of column indices for R
ia	:I: array of row pointers for A
ja	:I: array of column indices for A
ip	:I: array of row pointers for P
jp	:I: array of column indices for P
nin	:I: number of rows in R
ncin	:I: number of columns in R
iac	:O: array of row pointers for Ac
jac	:O: array of column indices for Ac
maxrout	:O: the maximum nonzeroes per row for R

Note

Computes the sparsity pattern of R∗A∗P. maxrout is output and is the maximum nonzeroes per row for r. On output we also have is iac (if jac is null) and jac (if jac entry is not null). R is (nc,n) A is (n,n) and P is (n,nc)!

Modified by Chensong Zhang on 09/11/2012

Definition at line 514 of file BlaSparseUtil.c.

9.33.2.10 fasp_sparse_wta_()

```
void fasp_sparse_wta_ (
            INT * jw,
            REAL * w,
```

```
            INT * ia,
            INT * ja,
            REAL * a,
            INT * nwp,
            INT * map,
            INT * jv,
            REAL * v,
            INT * nvp )
```

Calculate v^t = w^t A, where w is a sparse vector and A is sparse matrix. v is an array of dimension = number of columns in A.

Note

:I: is input :O: is output :IO: is both

Parameters

jw	:I: indices such that w[jw] is nonzero
w	:I: the values of w
ia	:I: array of row pointers for A
ja	:I: array of column indices for A
a	:I: entries of A
nwp	:I: number of nonzeroes in w (the length of w)
map	:I: number of columns in A
jv	:O: indices such that v[jv] is nonzero
v	:O: the result v^t=w^t A
nvp	:I: number of nonzeroes in v

Definition at line 648 of file BlaSparseUtil.c.

9.33.2.11 fasp_sparse_wtams_()

```
void fasp_sparse_wtams_ (
            INT * jw,
            INT * ia,
            INT * ja,
            INT * nwp,
            INT * map,
            INT * jv,
            INT * nvp,
            INT * icp )
```

Finds the nonzeroes in the result of v^t = w^t A, where w is a sparse vector and A is sparse matrix. jv is an integer array containing the indices of the nonzero elements in the result.

:I: is input :O: is output :IO: is both

Parameters

jw	:I: indices such that w[jw] is nonzero
ia	:I: array of row pointers for A
ja	:I: array of column indices for A
nwp	:I: number of nonzeroes in w (the length of w)
map	:I: number of columns in A
jv	:O: indices such that v[jv] is nonzero
nvp	:I: number of nonzeroes in v
icp	:IO: is a working array of length (∗map) which on output satisfies icp[jv[k]-1]=k; Values of icp[] at positions ∗ other than (jv[k]-1) remain unchanged.

Modified by Chensong Zhang on 09/11/2012

Definition at line 595 of file BlaSparseUtil.c.

9.33.2.12 fasp_sparse_ytx_()

```
void fasp_sparse_ytx_ (
            INT * jy,
            REAL * y,
            INT * jx,
            REAL * x,
            INT * nyp,
            INT * nxp,
            INT * icp,
            REAL * s )
```

Calculates s = y^t x. y is sparse, x is sparse.

note :I: is input :O: is output :IO: is both

Parameters

jy	:I: indices such that y[jy] is nonzero
y	:I: is a sparse vector.
nyp	:I: number of nonzeroes in y
jx	:I: indices such that x[jx] is nonzero
x	:I: is a sparse vector.
nxp	:I: number of nonzeroes in x
icp	???
s	:O: s = y^t x.

Definition at line 733 of file BlaSparseUtil.c.

9.33.2.13 fasp_sparse_ytxbig_()

```
void fasp_sparse_ytxbig_ (
            INT * jy,
            REAL * y,
            INT * nyp,
            REAL * x,
            REAL * s )
```

Calculates s = y^t x. y-sparse, x - no.

Note

:I: is input :O: is output :IO: is both

Parameters

jy	:I: indices such that y[jy] is nonzero
y	:I: is a sparse vector.
nyp	:I: number of nonzeroes in v
x	:I: also a vector assumed to have entry for any j=jy[i]-1; for i=1:nyp. This means that x here does not have to be sparse.
s	:O: s = y^t x.

Definition at line 699 of file BlaSparseUtil.c.

9.34 BlaSpmvBLC.c File Reference

Linear algebraic operations for dBLCmat matrices.

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

Functions

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

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

  Matrix-vector multiplication y = A∗x.

9.34.1 Detailed Description

Linear algebraic operations for dBLCmat matrices.

Note

This file contains Level-1 (Bla) functions. It requires: BlaSpmvCSR.c

9.34.2 Function Documentation

9.34.2.1 fasp_blas_dblc_aAxpy()

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

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

Parameters

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

Author

Xiaozhe Hu

Date

06/04/2010

Definition at line 38 of file BlaSpmvBLC.c.

9.34.2.2 fasp_blas_dblc_mxv()

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

Matrix-vector multiplication y = A∗x.

Parameters

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

Author

Chensong Zhang

Date

04/27/2013

Definition at line 162 of file BlaSpmvBLC.c.

9.35 BlaSpmvBSR.c File Reference

Linear algebraic operations for dBSRmat matrices.

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

Functions

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

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

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

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

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

Compute y := A∗x.

- void fasp_blas_dbsr_mxv_agg (const dBSRmat ∗A, const REAL ∗x, REAL ∗y)

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

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

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

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

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

9.35.1 Detailed Description

Linear algebraic operations for dBSRmat matrices.

Note

This file contains Level-1 (Bla) functions. It requires: AuxArray.c, AuxMemory.c, AuxThreads.c, BlaSmallMat.c, and BlaArray.c

9.35.2 Function Documentation

9.35.2.1 fasp_blas_dbsr_aAxpby()

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

Compute y := alpha∗A∗x + beta∗y.

Parameters

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

Author

Zhiyang Zhou

Date

10/25/2010

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

Note

Works for general nb (Xiaozhe)

Definition at line 67 of file BlaSpmvBSR.c.

9.35.2.2 fasp_blas_dbsr_aAxpy()

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

Compute y := alpha∗A∗x + y.

Parameters

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

Author

Zhiyang Zhou

Date

10/25/2010

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

Note

Works for general nb (Xiaozhe)

Definition at line 348 of file BlaSpmvBSR.c.

9.35.2.3 fasp_blas_dbsr_aAxpy_agg()

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

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

Parameters

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

Author

Xiaozhe Hu

Date

01/02/2014

Note

Works for general nb (Xiaozhe)

Definition at line 624 of file BlaSpmvBSR.c.

9.35.2.4 fasp_blas_dbsr_axm()

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

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

Parameters

A	Pointer to dBSRmat matrix A
alpha	REAL factor alpha

Author

Xiaozhe Hu

Date

05/26/2014

Definition at line 38 of file BlaSpmvBSR.c.

9.35.2.5 fasp_blas_dbsr_mxm()

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

Sparse matrix multiplication C=A∗B.

Parameters

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

Author

Xiaozhe Hu

Date

05/26/2014

Note

This fct will be replaced! – Xiaozhe

Definition at line 4646 of file BlaSpmvBSR.c.

9.35.2.6 fasp_blas_dbsr_mxv()

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

Compute y := A∗x.

Parameters

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

Author

Zhiyang Zhou

Date

10/25/2010

Note

Works for general nb (Xiaozhe)

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

Definition at line 910 of file BlaSpmvBSR.c.

9.35.2.7 fasp_blas_dbsr_mxv_agg()

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

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

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 2697 of file BlaSpmvBSR.c.

9.35.2.8 fasp_blas_dbsr_rap()

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

dBSRmat sparse matrix multiplication B=R∗A∗P

Parameters

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

Author

Xiaozhe Hu, Chunsheng Feng, Zheng Li

Date

10/24/2012

Note

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

Definition at line 4956 of file BlaSpmvBSR.c.

9.35.2.9 fasp_blas_dbsr_rap1()

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

dBSRmat sparse matrix multiplication B=R∗A∗P

Parameters

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

Author

Chunsheng Feng, Xiaoqiang Yue and Xiaozhe Hu

Date

08/08/2011

Note

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

Definition at line 4767 of file BlaSpmvBSR.c.

9.35.2.10 fasp_blas_dbsr_rap_agg()

```
void fasp_blas_dbsr_rap_agg (
            const dBSRmat * R,
            const dBSRmat * A,
            const 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

Definition at line 5221 of file BlaSpmvBSR.c.

9.36 BlaSpmvCSR.c File Reference

Linear algebraic operations for dCSRmat matrices.

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

Functions

- SHORT fasp_blas_dcsr_add (const dCSRmat ∗A, const REAL alpha, const 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 (const dCSRmat ∗A, const REAL ∗x, REAL ∗y)

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

  Matrix-vector multiplication y = A∗x (nonzeros of A = 1)
- void fasp_blas_dcsr_aAxpy (const REAL alpha, const dCSRmat ∗A, const REAL ∗x, REAL ∗y)

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

  Matrix-vector multiplication y = alpha∗A∗x + y (nonzeros of A = 1)
- REAL fasp_blas_dcsr_vmv (const dCSRmat ∗A, const REAL ∗x, const REAL ∗y)

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

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

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

  Triple sparse matrix multiplication B=R∗A∗P (nonzeros of R, P = 1)
- void fasp_blas_dcsr_rap_agg1 (const dCSRmat ∗R, const dCSRmat ∗A, const dCSRmat ∗P, dCSRmat ∗B)

  Triple sparse matrix multiplication B=R∗A∗P (nonzeros of R, P = 1)
- void fasp_blas_dcsr_ptap (const dCSRmat ∗Pt, const dCSRmat ∗A, const dCSRmat ∗P, dCSRmat ∗Ac)

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

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

9.36.1 Detailed Description

Linear algebraic operations for dCSRmat matrices.

Note

This file contains Level-1 (Bla) functions. It requires: AuxArray.c, AuxMemory.c, AuxThreads.c, BlaSparseCSR.c, BlaSparseUtil.c, and BlaArray.c
Sparse functions usually contain three runs. The three runs are all the same but thy serve different purpose.

Example: If you do c=a+b:

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

9.36.2 Function Documentation

9.36.2.1 fasp_blas_dcsr_aAxpy()

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

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

Parameters

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

Author

Chensong Zhang

Date

07/01/2009

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

Definition at line 486 of file BlaSpmvCSR.c.

9.36.2.2 fasp_blas_dcsr_aAxpy_agg()

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

Matrix-vector multiplication y = alpha∗A∗x + y (nonzeros of A = 1)

Parameters

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

Author

Xiaozhe Hu

Date

02/22/2011

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

Definition at line 601 of file BlaSpmvCSR.c.

9.36.2.3 fasp_blas_dcsr_add()

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

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

Parameters

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

Returns

FASP_SUCCESS if succeed, ERROR if not

Author

Xiaozhe Hu

Date

11/07/2009

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

Definition at line 57 of file BlaSpmvCSR.c.

9.36.2.4 fasp_blas_dcsr_axm()

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

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

Parameters

A	Pointer to dCSRmat matrix A
alpha	REAL factor alpha

Author

Chensong Zhang

Date

07/01/2009

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

Definition at line 209 of file BlaSpmvCSR.c.

9.36.2.5 fasp_blas_dcsr_mxm()

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

Sparse matrix multiplication C=A∗B.

Parameters

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

Author

Xiaozhe Hu

Date

11/07/2009

Warning

This fct will be replaced! –Chensong

Definition at line 767 of file BlaSpmvCSR.c.

9.36.2.6 fasp_blas_dcsr_mxv()

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

Matrix-vector multiplication y = A∗x.

Parameters

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

Author

Chensong Zhang

Date

07/01/2009

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

Definition at line 232 of file BlaSpmvCSR.c.

9.36.2.7 fasp_blas_dcsr_mxv_agg()

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

Matrix-vector multiplication y = A∗x (nonzeros of A = 1)

Parameters

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

Author

Xiaozhe Hu

Date

02/22/2011

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

Definition at line 429 of file BlaSpmvCSR.c.

9.36.2.8 fasp_blas_dcsr_ptap()

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

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

Parameters

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

Author

Ludmil Zikatanov, Chensong Zhang

Date

05/10/2010

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

Note

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

Definition at line 1607 of file BlaSpmvCSR.c.

9.36.2.9 fasp_blas_dcsr_rap()

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

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

Parameters

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

Author

Xuehai Huang, Chensong Zhang

Date

05/10/2010

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

Note

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

Definition at line 875 of file BlaSpmvCSR.c.

9.36.2.10 fasp_blas_dcsr_rap2()

```
dCSRmat fasp_blas_dcsr_rap2 (
            INT * ir,
            INT * jr,
            REAL * r,
            INT * ia,
            INT * ja,
            REAL * a,
            INT * ipt,
            INT * jpt,
            REAL * pt,
            INT n,
            INT nc,
            INT * maxrpout,
            INT * ipin,
            INT * jpin )
```

Compute R∗A∗P.

Author

Ludmil Zikatanov

Date

04/08/2010

Note

It uses dCSRmat only. The functions called from here are in sparse_util.c. Not used for the moment!

Definition at line 1707 of file BlaSpmvCSR.c.

9.36.2.11 fasp_blas_dcsr_rap4()

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

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

Parameters

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

Author

Feng Chunsheng, Yue Xiaoqiang

Date

08/02/2011

Note

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

Definition at line 1805 of file BlaSpmvCSR.c.

9.36.2.12 fasp_blas_dcsr_rap_agg()

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

Triple sparse matrix multiplication B=R∗A∗P (nonzeros of R, P = 1)

Parameters

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

Author

Xiaozhe Hu

Date

05/10/2010

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

Definition at line 1155 of file BlaSpmvCSR.c.

9.36.2.13 fasp_blas_dcsr_rap_agg1()

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

Triple sparse matrix multiplication B=R∗A∗P (nonzeros of R, P = 1)

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 1421 of file BlaSpmvCSR.c.

9.36.2.14 fasp_blas_dcsr_vmv()

```
REAL fasp_blas_dcsr_vmv (
            const dCSRmat * A,
            const REAL * x,
            const 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 712 of file BlaSpmvCSR.c.

9.37 BlaSpmvCSRL.c File Reference

Linear algebraic operations for dCSRLmat matrices.

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

Functions

- void fasp_blas_dcsrl_mxv (const dCSRLmat *A, const REAL *x, REAL *y)

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

9.37.1 Detailed Description

Linear algebraic operations for dCSRLmat matrices.

Note

This file contains Level-1 (Bla) functions.

Reference: John Mellor-Crummey and John Garvin Optimizaing sparse matrix vector product computations using unroll and jam, Tech Report Rice Univ, Aug 2002.

9.37.2 Function Documentation

9.37.2.1 fasp_blas_dcsrl_mxv()

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

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

Parameters

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

Author

Zhiyang Zhou, Chensong Zhang

Date

2011/01/07

Definition at line 36 of file BlaSpmvCSRL.c.

9.38 BlaSpmvSTR.c File Reference

Linear algebraic operations for dSTRmat matrices.

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

Functions

- void fasp_blas_dstr_aAxpy (const REAL alpha, const dSTRmat *A, const REAL *x, REAL *y)
  *Matrix-vector multiplication y = alpha*A*x + y.*
- void fasp_blas_dstr_mxv (const dSTRmat *A, const REAL *x, REAL *y)
  *Matrix-vector multiplication y = A*x.*
- INT fasp_blas_dstr_diagscale (const dSTRmat *A, dSTRmat *B)
  B=D^{-1}A.

9.38.1 Detailed Description

Linear algebraic operations for dSTRmat matrices.

Note

This file contains Level-1 (Bla) functions. It requires: AuxArray.c, AuxMemory.c, AuxThreads.c, BlaSmallMatInv.c, BlaSmallMat.c, and BlaSparseSTR.c

9.38.2 Function Documentation

9.38.2.1 fasp_blas_dstr_aAxpy()

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

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

Parameters

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

Author

Zhiyang Zhou, Xiaozhe Hu, Shiquan Zhang

Date

2010/10/15

Definition at line 61 of file BlaSpmvSTR.c.

9.38.2.2 fasp_blas_dstr_diagscale()

```
INT fasp_blas_dstr_diagscale (
            const dSTRmat * A,
            dSTRmat * B )
```

B=D^{-1}A.

Parameters

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

Author

Shiquan Zhang

Date

2010/10/15

Modified by Chunsheng Feng, Zheng Li on 08/30/2012

Definition at line 155 of file BlaSpmvSTR.c.

9.38.2.3 fasp_blas_dstr_mxv()

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

Matrix-vector multiplication y = A∗x.

Parameters

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

Author

Chensong Zhang

Date

04/27/2013

Definition at line 131 of file BlaSpmvSTR.c.

9.39 BlaVector.c File Reference

BLAS1 operations for vectors.

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

Functions

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

9.39.1 Detailed Description

BLAS1 operations for vectors.

Note

This file contains Level-1 (Bla) functions. It requires: AuxMessage.c, AuxThreads.c, and BlaArray.c

9.39.2 Function Documentation

9.39.2.1 fasp_blas_dvec_axpy()

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

y = a∗x + y

Parameters

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

Author

Chensong Zhang

Date

07/01/2009

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

Definition at line 41 of file BlaVector.c.

9.39.2.2 fasp_blas_dvec_axpyz()

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

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

Parameters

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

Author

Chensong Zhang

Date

07/01/2009

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

Definition at line 96 of file BlaVector.c.

9.39.2.3 fasp_blas_dvec_dotprod()

```
REAL fasp_blas_dvec_dotprod (
            const dvector * x,
            const 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

07/01/2009

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

Definition at line 236 of file BlaVector.c.

9.39.2.4 fasp_blas_dvec_norm1()

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

L1 norm of dvector x.

Parameters

x	Pointer to dvector x

Returns

L1 norm of x

Author

Chensong Zhang

Date

07/01/2009

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

Definition at line 130 of file BlaVector.c.

9.39.2.5 fasp_blas_dvec_norm2()

```
REAL fasp_blas_dvec_norm2 (
            const 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

Definition at line 170 of file BlaVector.c.

9.39.2.6 fasp_blas_dvec_norminf()

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

Linf norm of dvector x.

Parameters

x	Pointer to dvector x

Returns

L_inf norm of x

Author

Chensong Zhang

Date

07/01/2009

Definition at line 208 of file BlaVector.c.

9.39.2.7 fasp_blas_dvec_relerr()

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

Relative difference between two dvector x and y.

Parameters

x	Pointer to dvector x
y	Pointer to dvector y

Returns

Relative difference ||x-y||/||x||

Author

Chensong Zhang

Date

07/01/2009

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

Definition at line 278 of file BlaVector.c.

9.40 doxygen.h File Reference

Main page for Doygen documentation.

9.40.1 Detailed Description

Main page for Doygen documentation.

9.41 fasp.h File Reference

Main header file for the FASP project.

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

Data Structures

- struct ddenmat

    Dense matrix of REAL type.
- struct idenmat

    Dense matrix of INT type.
- struct dCSRmat

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

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

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

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

    Sparse matrix of REAL type in CSRL format.
- struct dSTRmat

    Structure matrix of REAL type.
- struct dvector

    Vector with n entries of REAL type.
- struct ivector

    Vector with n entries of INT type.
- struct ITS_param

    Parameters passed to iterative solvers.
- struct ILU_param

    Parameters for ILU.
- struct SWZ_param

    Parameters for Schwarz method.
- struct AMG_param

    Parameters for AMG solver.
- struct Mumps_data

    Parameters for MUMPS interface.
- struct Pardiso_data

    Parameters for Intel MKL PARDISO interface.

- struct SWZ_data

  Data for Schwarz methods.
- struct ILU_data

  Data for ILU setup.
- 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_diag_str

  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.

Macros

- #define __FASP_HEADER__
- #define FASP_VERSION 1.9

  FASP base version information.
- #define DLMALLOC OFF

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

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

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

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

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

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

Typedefs

- typedef struct ddenmat ddenmat
- typedef struct idenmat idenmat
- typedef struct dCSRmat dCSRmat
- typedef struct iCSRmat iCSRmat
- typedef struct dCOOmat dCOOmat
- typedef struct iCOOmat iCOOmat
- typedef struct dCSRLmat dCSRLmat
- typedef struct dSTRmat dSTRmat
- typedef struct dvector dvector
- typedef struct ivector ivector

Variables

- unsigned INT total_alloc_mem
- unsigned INT total_alloc_count

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

9.41.1 Detailed Description

Main header file for the FASP project.

Note

This header file contains general constants and data structures of FASP. It contains macros and data structure definitions; should not include function declarations here.

9.41.2 Macro Definition Documentation

9.41.2.1 __FASP_HEADER__

```
#define __FASP_HEADER__
```

indicate fasp.h has been included before

Definition at line 32 of file fasp.h.

9.41.2.2 ABS

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

absolute value of a

Definition at line 73 of file fasp.h.

9.41.2.3 DIAGONAL_PREF

```
#define DIAGONAL_PREF OFF
```

order each row such that diagonal appears first

Definition at line 57 of file fasp.h.

9.41.2.4 DLMALLOC

```
#define DLMALLOC OFF
```

For external software package support.

use dlmalloc instead of standard malloc

Definition at line 46 of file fasp.h.

9.41.2.5 FASP_GSRB

```
#define FASP_GSRB 1
```

MG level 0 use RedBlack Gauss Seidel Smoothing

Definition at line 1162 of file fasp.h.

9.41.2.6 FASP_VERSION

```
#define FASP_VERSION 1.9
```

FASP base version information.

faspsolver version

Definition at line 41 of file fasp.h.

9.41.2.7 GE

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

is a >= b?

Definition at line 79 of file fasp.h.

9.41.2.8 GT

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

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

is a > b?

Definition at line 78 of file fasp.h.

9.41.2.9 INT

```
#define INT int
```

regular integer type: int or long

Definition at line 63 of file fasp.h.

9.41.2.10 ISNAN

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

is a == NAN?

Definition at line 82 of file fasp.h.

9.41.2.11 LE

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

is a <= b?

Definition at line 81 of file fasp.h.

9.41.2.12 LONG

```
#define LONG long
```

long integer type

Definition at line 64 of file fasp.h.

9.41.2.13 LONGLONG

```
#define LONGLONG long long
```

long integer type

Definition at line 65 of file fasp.h.

9.41.2.14 LS

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

is a $<$ b?

Definition at line 80 of file fasp.h.

9.41.2.15 MAX

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

Definition of max, min, abs.

bigger one in a and b

Definition at line 71 of file fasp.h.

9.41.2.16 MIN

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

smaller one in a and b

Definition at line 72 of file fasp.h.

9.41.2.17 NEDMALLOC

```
#define NEDMALLOC OFF
```

use nedmalloc instead of standard malloc

Definition at line 47 of file fasp.h.

9.41.2.18 PUT_INT

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

Definition of print command in DEBUG mode.

print integer

Definition at line 87 of file fasp.h.

9.41.2.19 PUT_REAL

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

print real num

Definition at line 88 of file fasp.h.

9.41.2.20 REAL

```
#define REAL double
```

float type

Definition at line 66 of file fasp.h.

9.41.2.21 RS_C1

```
#define RS_C1 ON
```

Flags for internal uses.

Warning

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

Definition at line 55 of file fasp.h.

9.41.2.22 SHORT

```
#define SHORT short
```

FASP integer and floating point numbers.

short integer type

Definition at line 62 of file fasp.h.

9.41.3 Typedef Documentation

9.41.3.1 dCOOmat

```
typedef struct dCOOmat dCOOmat
```

Sparse matrix of REAL type in COO format

9.41.3.2 dCSRLmat

```
typedef struct dCSRLmat dCSRLmat
```

Sparse matrix of REAL type in CSRL format

9.41.3.3 dCSRmat

```
typedef struct dCSRmat dCSRmat
```

Sparse matrix of REAL type in CSR format

9.41.3.4 ddenmat

```
typedef struct ddenmat ddenmat
```

Dense matrix of REAL type

9.41.3.5 dSTRmat

```
typedef struct dSTRmat dSTRmat
```

Structured matrix of REAL type

9.41.3.6 dvector

```
typedef struct dvector dvector
```

Vector of REAL type

9.41.3.7 iCOOmat

```
typedef struct iCOOmat iCOOmat
```

Sparse matrix of INT type in COO format

9.41.3.8 iCSRmat

```
typedef struct iCSRmat iCSRmat
```

Sparse matrix of INT type in CSR format

9.41.3.9 idenmat

```
typedef struct idenmat idenmat
```

Dense matrix of INT type

9.41.3.10 ivector

```
typedef struct ivector ivector
```

Vector of INT type

9.41.4 Variable Documentation

9.41.4.1 count

```
INT count
```

Counter for multiple calls

9.41.4.2 IMAP

```
INT* IMAP
```

Red Black Gs Smoother imap

9.41.4.3 MAXIMAP

```
INT MAXIMAP
```

Red Black Gs Smoother max DOFs of reservoir

9.41.4.4 nx_rb

```
INT nx_rb
```

Red Black Gs Smoother Nx

9.41.4.5 ny_rb

```
INT ny_rb
```

Red Black Gs Smoother Ny

9.41.4.6 nz_rb

```
INT nz_rb
```

Red Black Gs Smoother Nz

9.41.4.7 total_alloc_count

```
unsigned INT total_alloc_count
```

Total allocated memory amount.

total allocation times

Definition at line 44 of file AuxMemory.c.

9.41.4.8 total_alloc_mem

```
unsigned INT total_alloc_mem
```

total allocated memory

Definition at line 43 of file AuxMemory.c.

9.42 fasp_block.h File Reference

Header file for FASP block matrices.

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

Data Structures

- struct dBSRmat

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

  Block REAL CSR matrix format.
- struct iBLCmat

  Block INT CSR matrix format.
- struct block_dvector

  Block REAL vector structure.
- struct block_ivector

  Block INT vector structure.
- struct AMG_data_bsr

  Data for multigrid levels. (BSR format)
- struct precond_diagbsr

  Data passed to diagnal preconditioner for dBSRmat matrices.
- struct precond_data_bsr

  Data passed to the preconditioners.
- struct precond_block_data

  Data passed to the preconditioner for block preconditioning for dBLCmat format.
- struct precond_sweeping_data

  Data passed to the preconditioner for sweeping preconditioning.

Macros

- #define __FASPBLOCK_HEADER__

Typedefs

- typedef struct dBSRmat dBSRmat
- typedef struct dBLCmat dBLCmat
- typedef struct iBLCmat iBLCmat
- typedef struct block_dvector block_dvector
- typedef struct block_ivector block_ivector

9.42.1 Detailed Description

Header file for FASP block matrices.

Note

This header file contains definitions of block matrices, including grid-major type and variable-major type. In this header, we only define macros and data structures, not function declarations.

9.42.2 Macro Definition Documentation

9.42.2.1 __FASPBLOCK_HEADER__

```
#define __FASPBLOCK_HEADER__
```

indicate fasp_block.h has been included before

Definition at line 18 of file fasp_block.h.

9.42.3 Typedef Documentation

9.42.3.1 block_dvector

```
typedef struct block_dvector block_dvector
```

Vector of REAL type in Block format

9.42.3.2 block_ivector

```
typedef struct block_ivector block_ivector
```

Vector of INT type in Block format

9.42.3.3 dBLCmat

```
typedef struct dBLCmat dBLCmat
```

Matrix of REAL type in Block CSR format

9.42.3.4 dBSRmat

```
typedef struct dBSRmat dBSRmat
```

Matrix of REAL type in BSR format

9.42.3.5 iBLCmat

```
typedef struct iBLCmat iBLCmat
```

Matrix of INT type in Block CSR format

9.43 fasp_const.h File Reference

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

Macros

- #define FASP_SUCCESS 0
  Definition of return status and error messages.
- #define ERROR_OPEN_FILE -10
- #define ERROR_WRONG_FILE -11
- #define ERROR_INPUT_PAR -13
- #define ERROR_REGRESS -14
- #define ERROR_MAT_SIZE -15
- #define ERROR_NUM_BLOCKS -18
- #define ERROR_MISC -19
- #define ERROR_ALLOC_MEM -20
- #define ERROR_DATA_STRUCTURE -21
- #define ERROR_DATA_ZERODIAG -22
- #define ERROR_DUMMY_VAR -23

- #define ERROR_AMG_INTERP_TYPE -30
- #define ERROR_AMG_SMOOTH_TYPE -31
- #define ERROR_AMG_COARSE_TYPE -32
- #define ERROR_AMG_COARSEING -33
- #define ERROR_SOLVER_TYPE -40
- #define ERROR_SOLVER_PRECTYPE -41
- #define ERROR_SOLVER_STAG -42
- #define ERROR_SOLVER_SOLSTAG -43
- #define ERROR_SOLVER_TOLSMALL -44
- #define ERROR_SOLVER_ILUSETUP -45
- #define ERROR_SOLVER_MISC -46
- #define ERROR_SOLVER_MAXIT -48
- #define ERROR_SOLVER_EXIT -49
- #define ERROR_QUAD_TYPE -60
- #define ERROR_QUAD_DIM -61
- #define ERROR_LIC_TYPE -80
- #define ERROR_UNKNOWN -99
- #define TRUE 1

  Definition of logic type.
- #define FALSE 0
- #define ON 1

  Definition of switch.
- #define OFF 0
- #define PRINT_NONE 0

  Print level for all subroutines – not including DEBUG output.
- #define PRINT_MIN 1
- #define PRINT_SOME 2
- #define PRINT_MORE 4
- #define PRINT_MOST 8
- #define PRINT_ALL 10
- #define MAT_FREE 0

  Definition of matrix format.
- #define MAT_CSR 1
- #define MAT_BSR 2
- #define MAT_STR 3
- #define MAT_CSRL 6
- #define MAT_SymCSR 7
- #define MAT_BLC 8
- #define MAT_bCSR 11
- #define MAT_bBSR 12
- #define MAT_bSTR 13
- #define SOLVER_DEFAULT 0

  Definition of solver types for iterative methods.
- #define SOLVER_CG 1
- #define SOLVER_BiCGstab 2
- #define SOLVER_VBiCGstab 9
- #define SOLVER_MinRes 3
- #define SOLVER_GMRES 4
- #define SOLVER_VGMRES 5
- #define SOLVER_VFGMRES 6

- #define SOLVER_GCG 7
- #define SOLVER_GCR 8
- #define SOLVER_SCG 11
- #define SOLVER_SBiCGstab 12
- #define SOLVER_SMinRes 13
- #define SOLVER_SGMRES 14
- #define SOLVER_SVGMRES 15
- #define SOLVER_SVFGMRES 16
- #define SOLVER_SGCG 17
- #define SOLVER_AMG 21
- #define SOLVER_FMG 22
- #define SOLVER_SUPERLU 31
- #define SOLVER_UMFPACK 32
- #define SOLVER_MUMPS 33
- #define SOLVER_PARDISO 34
- #define STOP_REL_RES 1

  Definition of iterative solver stopping criteria types.
- #define STOP_REL_PRECRES 2
- #define STOP_MOD_REL_RES 3
- #define PREC_NULL 0

  Definition of preconditioner type for iterative methods.
- #define PREC_DIAG 1
- #define PREC_AMG 2
- #define PREC_FMG 3
- #define PREC_ILU 4
- #define PREC_SCHWARZ 5
- #define ILUk 1

  Type of ILU methods.
- #define ILUt 2
- #define ILUtp 3
- #define SCHWARZ_FORWARD 1

  Type of Schwarz smoother.
- #define SCHWARZ_BACKWARD 2
- #define SCHWARZ_SYMMETRIC 3
- #define CLASSIC_AMG 1

  Definition of AMG types.
- #define SA_AMG 2
- #define UA_AMG 3
- #define PAIRWISE 1

  Definition of aggregation types.
- #define VMB 2
- #define V_CYCLE 1

  Definition of cycle types.
- #define W_CYCLE 2
- #define AMLI_CYCLE 3
- #define NL_AMLI_CYCLE 4
- #define SMOOTHER_JACOBI 1

  Definition of standard smoother types.
- #define SMOOTHER_GS 2

- #define SMOOTHER_SGS 3
- #define SMOOTHER_CG 4
- #define SMOOTHER_SOR 5
- #define SMOOTHER_SSOR 6
- #define SMOOTHER_GSOR 7
- #define SMOOTHER_SGSOR 8
- #define SMOOTHER_POLY 9
- #define SMOOTHER_L1DIAG 10
- #define SMOOTHER_BLKOIL 11

    Definition of specialized smoother types.
- #define SMOOTHER_SPETEN 19
- #define COARSE_RS 1

    Definition of coarsening types.
- #define COARSE_RSP 2
- #define COARSE_CR 3
- #define COARSE_AC 4
- #define COARSE_MIS 5
- #define INTERP_DIR 1

    Definition of interpolation types.
- #define INTERP_STD 2
- #define INTERP_ENG 3
- #define INTERP_EXT 6
- #define G0PT -5

    Type of vertices (DOFs) for coarsening.
- #define UNPT -1
- #define FGPT 0
- #define CGPT 1
- #define ISPT 2
- #define NO_ORDER 0

    Definition of smoothing order.
- #define CF_ORDER 1
- #define ILU_MC_OMP 1
- #define USERDEFINED 0

    Type of ordering for smoothers.
- #define CPFIRST 1
- #define FPFIRST -1
- #define ASCEND 12
- #define DESCEND 21
- #define BIGREAL 1e+20

    Some global constants.
- #define SMALLREAL 1e-20
- #define SMALLREAL2 1e-40
- #define MAX_REFINE_LVL 20
- #define MAX_AMG_LVL 20
- #define MIN_CDOF 20
- #define MIN_CRATE 0.9
- #define MAX_CRATE 20.0
- #define MAX_RESTART 20
- #define MAX_STAG 20
- #define STAG_RATIO 1e-4
- #define OPENMP_HOLDS 2000

9.43.1 Detailed Description

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

Warning

This is for internal use only. Do NOT change!

9.43.2 Macro Definition Documentation

9.43.2.1 AMLI_CYCLE

```
#define AMLI_CYCLE 3
```

AMLI-cycle

Definition at line 177 of file fasp_const.h.

9.43.2.2 ASCEND

```
#define ASCEND 12
```

Ascending order

Definition at line 239 of file fasp_const.h.

9.43.2.3 BIGREAL

```
#define BIGREAL 1e+20
```

Some global constants.

A large real number

Definition at line 245 of file fasp_const.h.

9.43.2.4 CF_ORDER

```
#define CF_ORDER 1
```

C/F order smoothing

Definition at line 230 of file fasp_const.h.

9.43.2.5 CGPT

```
#define CGPT 1
```

Coarse grid points

Definition at line 223 of file fasp_const.h.

9.43.2.6 CLASSIC_AMG

```
#define CLASSIC_AMG 1
```

Definition of AMG types.

classic AMG

Definition at line 162 of file fasp_const.h.

9.43.2.7 COARSE_AC

```
#define COARSE_AC 4
```

Aggressive coarsening

Definition at line 206 of file fasp_const.h.

9.43.2.8 COARSE_CR

```
#define COARSE_CR 3
```

Compatible relaxation

Definition at line 205 of file fasp_const.h.

9.43.2.9 COARSE_MIS

```
#define COARSE_MIS 5
```

Aggressive coarsening based on MIS

Definition at line 207 of file fasp_const.h.

9.43.2.10 COARSE_RS

```
#define COARSE_RS 1
```

Definition of coarsening types.

Classical

Definition at line 203 of file fasp_const.h.

9.43.2.11 COARSE_RSP

```
#define COARSE_RSP 2
```

Classical, with positive offdiags

Definition at line 204 of file fasp_const.h.

9.43.2.12 CPFIRST

```
#define CPFIRST 1
```

C-points first order

Definition at line 237 of file fasp_const.h.

9.43.2.13 DESCEND

```
#define DESCEND 21
```

Descending order

Definition at line 240 of file fasp_const.h.

9.43.2.14 ERROR_ALLOC_MEM

```
#define ERROR_ALLOC_MEM -20
```

fail to allocate memory

Definition at line 30 of file fasp_const.h.

9.43.2.15 ERROR_AMG_COARSE_TYPE

```
#define ERROR_AMG_COARSE_TYPE -32
```

unknown coarsening type

Definition at line 37 of file fasp_const.h.

9.43.2.16 ERROR_AMG_COARSEING

```
#define ERROR_AMG_COARSEING -33
```

coarsening step failed to complete

Definition at line 38 of file fasp_const.h.

9.43.2.17 ERROR_AMG_INTERP_TYPE

```
#define ERROR_AMG_INTERP_TYPE -30
```

unknown interpolation type

Definition at line 35 of file fasp_const.h.

9.43.2.18 ERROR_AMG_SMOOTH_TYPE

```
#define ERROR_AMG_SMOOTH_TYPE -31
```

unknown smoother type

Definition at line 36 of file fasp_const.h.

9.43.2.19 ERROR_DATA_STRUCTURE

```
#define ERROR_DATA_STRUCTURE -21
```

problem with data structures

Definition at line 31 of file fasp_const.h.

9.43.2.20 ERROR_DATA_ZERODIAG

```
#define ERROR_DATA_ZERODIAG -22
```

matrix has zero diagonal entries

Definition at line 32 of file fasp_const.h.

9.43.2.21 ERROR_DUMMY_VAR

```
#define ERROR_DUMMY_VAR -23
```

unexpected input data

Definition at line 33 of file fasp_const.h.

9.43.2.22 ERROR_INPUT_PAR

```
#define ERROR_INPUT_PAR -13
```

wrong input argument

Definition at line 24 of file fasp_const.h.

9.43.2.23 ERROR_LIC_TYPE

```
#define ERROR_LIC_TYPE -80
```

wrong license type

Definition at line 53 of file fasp_const.h.

9.43.2.24 ERROR_MAT_SIZE

```
#define ERROR_MAT_SIZE -15
```

wrong problem size

Definition at line 26 of file fasp_const.h.

9.43.2.25 ERROR_MISC

```
#define ERROR_MISC -19
```

other error

Definition at line 28 of file fasp_const.h.

9.43.2.26 ERROR_NUM_BLOCKS

```
#define ERROR_NUM_BLOCKS -18
```

wrong number of blocks

Definition at line 27 of file fasp_const.h.

9.43.2.27 ERROR_OPEN_FILE

```
#define ERROR_OPEN_FILE -10
```

fail to open a file

Definition at line 22 of file fasp_const.h.

9.43.2.28 ERROR_QUAD_DIM

```
#define ERROR_QUAD_DIM -61
```

unsupported quadrature dim

Definition at line 51 of file fasp_const.h.

9.43.2.29 ERROR_QUAD_TYPE

```
#define ERROR_QUAD_TYPE -60
```

unknown quadrature type

Definition at line 50 of file fasp_const.h.

9.43.2.30 ERROR_REGRESS

```
#define ERROR_REGRESS -14
```

regression test fail

Definition at line 25 of file fasp_const.h.

9.43.2.31 ERROR_SOLVER_EXIT

```
#define ERROR_SOLVER_EXIT -49
```

solver does not quit successfully

Definition at line 48 of file fasp_const.h.

9.43.2.32 ERROR_SOLVER_ILUSETUP

```
#define ERROR_SOLVER_ILUSETUP -45
```

ILU setup error

Definition at line 45 of file fasp_const.h.

9.43.2.33 ERROR_SOLVER_MAXIT

```
#define ERROR_SOLVER_MAXIT -48
```

maximal iteration number exceeded

Definition at line 47 of file fasp_const.h.

9.43.2.34 ERROR_SOLVER_MISC

```
#define ERROR_SOLVER_MISC -46
```

misc solver error during run time

Definition at line 46 of file fasp_const.h.

9.43.2.35 ERROR_SOLVER_PRECTYPE

```
#define ERROR_SOLVER_PRECTYPE -41
```

unknown precond type

Definition at line 41 of file fasp_const.h.

9.43.2.36 ERROR_SOLVER_SOLSTAG

```
#define ERROR_SOLVER_SOLSTAG -43
```

solver's solution is too small

Definition at line 43 of file fasp_const.h.

9.43.2.37 ERROR_SOLVER_STAG

```
#define ERROR_SOLVER_STAG -42
```

solver stagnates

Definition at line 42 of file fasp_const.h.

9.43.2.38 ERROR_SOLVER_TOLSMALL

```
#define ERROR_SOLVER_TOLSMALL -44
```

solver's tolerance is too small

Definition at line 44 of file fasp_const.h.

9.43.2.39 ERROR_SOLVER_TYPE

```
#define ERROR_SOLVER_TYPE -40
```

unknown solver type

Definition at line 40 of file fasp_const.h.

9.43.2.40 ERROR_UNKNOWN

```
#define ERROR_UNKNOWN -99
```

an unknown error type

Definition at line 55 of file fasp_const.h.

9.43.2.41 ERROR_WRONG_FILE

```
#define ERROR_WRONG_FILE -11
```

input contains wrong format

Definition at line 23 of file fasp_const.h.

9.43.2.42 FALSE

```
#define FALSE 0
```

logic FALSE

Definition at line 61 of file fasp_const.h.

9.43.2.43 FASP_SUCCESS

```
#define FASP_SUCCESS 0
```

Definition of return status and error messages.

return from function successfully

Definition at line 20 of file fasp_const.h.

9.43.2.44 FGPT

```
#define FGPT 0
```

Fine grid points

Definition at line 222 of file fasp_const.h.

9.43.2.45 FPFIRST

```
#define FPFIRST -1
```

F-points first order

Definition at line 238 of file fasp_const.h.

9.43.2.46 G0PT

```
#define G0PT -5
```

Type of vertices (DOFs) for coarsening.

Cannot fit in aggregates

Definition at line 220 of file fasp_const.h.

9.43.2.47 ILU_MC_OMP

```
#define ILU_MC_OMP 1
```

Multi-colors Parallel smoothing

Definition at line 231 of file fasp_const.h.

9.43.2.48 ILUk

```
#define ILUk 1
```

Type of ILU methods.

ILUk

Definition at line 148 of file fasp_const.h.

9.43.2.49 ILUt

```
#define ILUt 2
```

ILUt

Definition at line 149 of file fasp_const.h.

9.43.2.50 ILUtp

```
#define ILUtp 3
```

ILUtp

Definition at line 150 of file fasp_const.h.

9.43.2.51 INTERP_DIR

```
#define INTERP_DIR 1
```

Definition of interpolation types.

Direct interpolation

Definition at line 212 of file fasp_const.h.

9.43.2.52 INTERP_ENG

```
#define INTERP_ENG 3
```

Energy minimization interpolation

Definition at line 214 of file fasp_const.h.

9.43.2.53 INTERP_EXT

```
#define INTERP_EXT 6
```

Extended interpolation

Definition at line 215 of file fasp_const.h.

9.43.2.54 INTERP_STD

```
#define INTERP_STD 2
```

Standard interpolation

Definition at line 213 of file fasp_const.h.

9.43.2.55 ISPT

```
#define ISPT 2
```

Isolated points

Definition at line 224 of file fasp_const.h.

9.43.2.56 MAT_bBSR

```
#define MAT_bBSR 12
```

block BSR/CSR matrix

Definition at line 94 of file fasp_const.h.

9.43.2.57 MAT_bCSR

```
#define MAT_bCSR 11
```

block CSR/CSR matrix == 2*2 BLC matrix

Definition at line 93 of file fasp_const.h.

9.43.2.58 MAT_BLC

```
#define MAT_BLC 8
```

block CSR matrix

Definition at line 89 of file fasp_const.h.

9.43.2.59 MAT_BSR

```
#define MAT_BSR 2
```

block-wise compressed sparse row

Definition at line 85 of file fasp_const.h.

9.43.2.60 MAT_bSTR

```
#define MAT_bSTR 13
```

block STR/CSR matrix

Definition at line 95 of file fasp_const.h.

9.43.2.61 MAT_CSR

```
#define MAT_CSR 1
```

compressed sparse row

Definition at line 84 of file fasp_const.h.

9.43.2.62 MAT_CSRL

```
#define MAT_CSRL 6
```

modified CSR to reduce cache missing

Definition at line 87 of file fasp_const.h.

9.43.2.63 MAT_FREE

```
#define MAT_FREE 0
```

Definition of matrix format.

matrix-free format: only mxv action

Definition at line 82 of file fasp_const.h.

9.43.2.64 MAT_STR

```
#define MAT_STR 3
```

structured sparse matrix

Definition at line 86 of file fasp_const.h.

9.43.2.65 MAT_SymCSR

```
#define MAT_SymCSR 7
```

symmetric CSR format

Definition at line 88 of file fasp_const.h.

9.43.2.66 MAX_AMG_LVL

```
#define MAX_AMG_LVL 20
```

Maximal AMG coarsening level

Definition at line 249 of file fasp_const.h.

9.43.2.67 MAX_CRATE

```
#define MAX_CRATE 20.0
```

Maximal coarsening ratio

Definition at line 252 of file fasp_const.h.

9.43.2.68 MAX_REFINE_LVL

```
#define MAX_REFINE_LVL 20
```

Maximal refinement level

Definition at line 248 of file fasp_const.h.

9.43.2.69 MAX_RESTART

```
#define MAX_RESTART 20
```

Maximal restarting number

Definition at line 253 of file fasp_const.h.

9.43.2.70 MAX_STAG

```
#define MAX_STAG 20
```

Maximal number of stagnation times

Definition at line 254 of file fasp_const.h.

9.43.2.71 MIN_CDOF

```
#define MIN_CDOF 20
```

Minimal number of coarsest variables

Definition at line 250 of file fasp_const.h.

9.43.2.72 MIN_CRATE

```
#define MIN_CRATE 0.9
```

Minimal coarsening ratio

Definition at line 251 of file fasp_const.h.

9.43.2.73 NL_AMLI_CYCLE

```
#define NL_AMLI_CYCLE 4
```

Nonlinear AMLI-cycle

Definition at line 178 of file fasp_const.h.

9.43.2.74 NO_ORDER

```
#define NO_ORDER 0
```

Definition of smoothing order.

Natural order smoothing

Definition at line 229 of file fasp_const.h.

9.43.2.75 OFF

```
#define OFF 0
```

turn off certain parameter

Definition at line 67 of file fasp_const.h.

9.43.2.76 ON

```
#define ON 1
```

Definition of switch.

turn on certain parameter

Definition at line 66 of file fasp_const.h.

9.43.2.77 OPENMP_HOLDS

```
#define OPENMP_HOLDS 2000
```

Smallest size for OpenMP version

Definition at line 256 of file fasp_const.h.

9.43.2.78 PAIRWISE

```
#define PAIRWISE 1
```

Definition of aggregation types.

pairwise aggregation

Definition at line 169 of file fasp_const.h.

9.43.2.79 PREC_AMG

```
#define PREC_AMG 2
```

with AMG precond

Definition at line 140 of file fasp_const.h.

9.43.2.80 PREC_DIAG

```
#define PREC_DIAG 1
```

with diagonal precond

Definition at line 139 of file fasp_const.h.

9.43.2.81 PREC_FMG

```
#define PREC_FMG 3
```

with full AMG precond

Definition at line 141 of file fasp_const.h.

9.43.2.82 PREC_ILU

```
#define PREC_ILU 4
```

with ILU precond

Definition at line 142 of file fasp_const.h.

9.43.2.83 PREC_NULL

```
#define PREC_NULL 0
```

Definition of preconditioner type for iterative methods.

with no precond

Definition at line 138 of file fasp_const.h.

9.43.2.84 PREC_SCHWARZ

```
#define PREC_SCHWARZ 5
```

with Schwarz preconditioner

Definition at line 143 of file fasp_const.h.

9.43.2.85 PRINT_ALL

```
#define PRINT_ALL 10
```

all: all printouts, including files

Definition at line 77 of file fasp_const.h.

9.43.2.86 PRINT_MIN

```
#define PRINT_MIN 1
```

quiet: print error, important warnings

Definition at line 73 of file fasp_const.h.

9.43.2.87 PRINT_MORE

```
#define PRINT_MORE 4
```

more: print some useful debug info

Definition at line 75 of file fasp_const.h.

9.43.2.88 PRINT_MOST

```
#define PRINT_MOST 8
```

most: maximal printouts, no files

Definition at line 76 of file fasp_const.h.

9.43.2.89 PRINT_NONE

```
#define PRINT_NONE 0
```

Print level for all subroutines – not including DEBUG output.

silent: no printout at all

Definition at line 72 of file fasp_const.h.

9.43.2.90 PRINT_SOME

```
#define PRINT_SOME 2
```

some: print less important warnings

Definition at line 74 of file fasp_const.h.

9.43.2.91 SA_AMG

```
#define SA_AMG 2
```

smoothed aggregation AMG

Definition at line 163 of file fasp_const.h.

9.43.2.92 SCHWARZ_BACKWARD

```
#define SCHWARZ_BACKWARD 2
```

Backward ordering

Definition at line 156 of file fasp_const.h.

9.43.2.93 SCHWARZ_FORWARD

```
#define SCHWARZ_FORWARD 1
```

Type of Schwarz smoother.

Forward ordering

Definition at line 155 of file fasp_const.h.

9.43.2.94 SCHWARZ_SYMMETRIC

```
#define SCHWARZ_SYMMETRIC 3
```

Symmetric smoother

Definition at line 157 of file fasp_const.h.

9.43.2.95 SMALLREAL

```
#define SMALLREAL 1e-20
```

A small real number

Definition at line 246 of file fasp_const.h.

9.43.2.96 SMALLREAL2

```
#define SMALLREAL2 1e-40
```

An extremely small real number

Definition at line 247 of file fasp_const.h.

9.43.2.97 SMOOTHER_BLKOIL

```
#define SMOOTHER_BLKOIL 11
```

Definition of specialized smoother types.

Used in monolithic AMG for black-oil

Definition at line 197 of file fasp_const.h.

9.43.2.98 SMOOTHER_CG

```
#define SMOOTHER_CG 4
```

CG as a smoother

Definition at line 186 of file fasp_const.h.

9.43.2.99 SMOOTHER_GS

```
#define SMOOTHER_GS 2
```

Gauss-Seidel smoother

Definition at line 184 of file fasp_const.h.

9.43.2.100 SMOOTHER_GSOR

```
#define SMOOTHER_GSOR 7
```

GS + SOR smoother

Definition at line 189 of file fasp_const.h.

9.43.2.101 SMOOTHER_JACOBI

```
#define SMOOTHER_JACOBI 1
```

Definition of standard smoother types.

Jacobi smoother

Definition at line 183 of file fasp_const.h.

9.43.2.102 SMOOTHER_L1DIAG

```
#define SMOOTHER_L1DIAG 10
```

L1 norm diagonal scaling smoother

Definition at line 192 of file fasp_const.h.

9.43.2.103 SMOOTHER_POLY

```
#define SMOOTHER_POLY 9
```

Polynomial smoother

Definition at line 191 of file fasp_const.h.

9.43.2.104 SMOOTHER_SGS

```
#define SMOOTHER_SGS 3
```

Symmetric Gauss-Seidel smoother

Definition at line 185 of file fasp_const.h.

9.43.2.105 SMOOTHER_SGSOR

```
#define SMOOTHER_SGSOR 8
```

SGS + SSOR smoother

Definition at line 190 of file fasp_const.h.

9.43.2.106 SMOOTHER_SOR

```
#define SMOOTHER_SOR 5
```

SOR smoother

Definition at line 187 of file fasp_const.h.

9.43.2.107 SMOOTHER_SPETEN

```
#define SMOOTHER_SPETEN 19
```

Used in monolithic AMG for black-oil

Definition at line 198 of file fasp_const.h.

9.43.2.108 SMOOTHER_SSOR

```
#define SMOOTHER_SSOR 6
```

SSOR smoother

Definition at line 188 of file fasp_const.h.

9.43.2.109 SOLVER_AMG

```
#define SOLVER_AMG 21
```

AMG as an iterative solver

Definition at line 120 of file fasp_const.h.

9.43.2.110 SOLVER_BiCGstab

```
#define SOLVER_BiCGstab 2
```

Bi-Conjugate Gradient Stabilized

Definition at line 103 of file fasp_const.h.

9.43.2.111 SOLVER_CG

```
#define SOLVER_CG 1
```

Conjugate Gradient

Definition at line 102 of file fasp_const.h.

9.43.2.112 SOLVER_DEFAULT

```
#define SOLVER_DEFAULT 0
```

Definition of solver types for iterative methods.

Use default solver in FASP

Definition at line 100 of file fasp_const.h.

9.43.2.113 SOLVER_FMG

```
#define SOLVER_FMG 22
```

Full AMG as an solver

Definition at line 121 of file fasp_const.h.

9.43.2.114 SOLVER_GCG

```
#define SOLVER_GCG 7
```

Generalized Conjugate Gradient

Definition at line 109 of file fasp_const.h.

9.43.2.115 SOLVER_GCR

```
#define SOLVER_GCR 8
```

Generalized Conjugate Residual

Definition at line 110 of file fasp_const.h.

9.43.2.116 SOLVER_GMRES

```
#define SOLVER_GMRES 4
```

Generalized Minimal Residual

Definition at line 106 of file fasp_const.h.

9.43.2.117 SOLVER_MinRes

```
#define SOLVER_MinRes 3
```

Minimal Residual

Definition at line 105 of file fasp_const.h.

9.43.2.118 SOLVER_MUMPS

```
#define SOLVER_MUMPS 33
```

Direct Solver: MUMPS

Definition at line 125 of file fasp_const.h.

9.43.2.119 SOLVER_PARDISO

```
#define SOLVER_PARDISO 34
```

Direct Solver: PARDISO

Definition at line 126 of file fasp_const.h.

9.43.2.120 SOLVER_SBiCGstab

```
#define SOLVER_SBiCGstab 12
```

BiCGstab with safety net

Definition at line 113 of file fasp_const.h.

9.43.2.121 SOLVER_SCG

```
#define SOLVER_SCG 11
```

Conjugate Gradient with safety net

Definition at line 112 of file fasp_const.h.

9.43.2.122 SOLVER_SGCG

```
#define SOLVER_SGCG 17
```

GCG with safety net

Definition at line 118 of file fasp_const.h.

9.43.2.123 SOLVER_SGMRES

```
#define SOLVER_SGMRES 14
```

GMRes with safety net

Definition at line 115 of file fasp_const.h.

9.43.2.124 SOLVER_SMinRes

```
#define SOLVER_SMinRes 13
```

MinRes with safety net

Definition at line 114 of file fasp_const.h.

9.43.2.125 SOLVER_SUPERLU

```
#define SOLVER_SUPERLU 31
```

Direct Solver: SuperLU

Definition at line 123 of file fasp_const.h.

9.43.2.126 SOLVER_SVFGMRES

```
#define SOLVER_SVFGMRES 16
```

Variable-restart FGMRES with safety net

Definition at line 117 of file fasp_const.h.

9.43.2.127 SOLVER_SVGMRES

```
#define SOLVER_SVGMRES 15
```

Variable-restart GMRES with safety net

Definition at line 116 of file fasp_const.h.

9.43.2.128 SOLVER_UMFPACK

```
#define SOLVER_UMFPACK 32
```

Direct Solver: UMFPack

Definition at line 124 of file fasp_const.h.

9.43.2.129 SOLVER_VBiCGstab

```
#define SOLVER_VBiCGstab 9
```

VBi-Conjugate Gradient Stabilized

Definition at line 104 of file fasp_const.h.

9.43.2.130 SOLVER_VFGMRES

```
#define SOLVER_VFGMRES 6
```

Variable Restarting Flexible GMRES

Definition at line 108 of file fasp_const.h.

9.43.2.131 SOLVER_VGMRES

```
#define SOLVER_VGMRES 5
```

Variable Restarting GMRES

Definition at line 107 of file fasp_const.h.

9.43.2.132 STAG_RATIO

```
#define STAG_RATIO 1e-4
```

Stagnation tolerance = tol∗STAGRATIO

Definition at line 255 of file fasp_const.h.

9.43.2.133 STOP_MOD_REL_RES

```
#define STOP_MOD_REL_RES 3
```

modified relative residual ||r||/||x||

Definition at line 133 of file fasp_const.h.

9.43.2.134 STOP_REL_PRECRES

```
#define STOP_REL_PRECRES 2
```

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

Definition at line 132 of file fasp_const.h.

9.43.2.135 STOP_REL_RES

```
#define STOP_REL_RES 1
```

Definition of iterative solver stopping criteria types.

relative residual ||r||/||b||

Definition at line 131 of file fasp_const.h.

9.43.2.136 TRUE

```
#define TRUE 1
```

Definition of logic type.

logic TRUE

Definition at line 60 of file fasp_const.h.

9.43.2.137 UA_AMG

```
#define UA_AMG 3
```

unsmoothed aggregation AMG

Definition at line 164 of file fasp_const.h.

9.43.2.138 UNPT

```
#define UNPT -1
```

Undetermined points

Definition at line 221 of file fasp_const.h.

9.43.2.139 USERDEFINED

```
#define USERDEFINED 0
```

Type of ordering for smoothers.

User defined order

Definition at line 236 of file fasp_const.h.

9.43.2.140 V_CYCLE

```
#define V_CYCLE 1
```

Definition of cycle types.

V-cycle

Definition at line 175 of file fasp_const.h.

9.43.2.141 VMB

```
#define VMB 2
```

VMB aggregation

Definition at line 170 of file fasp_const.h.

9.43.2.142 W_CYCLE

```
#define W_CYCLE 2
```

W-cycle

Definition at line 176 of file fasp_const.h.

9.44 fasp_grid.h File Reference

Header file for FASP grid.

Data Structures

- struct grid2d

  Two dimensional grid data structure.

Macros

- #define __FASPGRID_HEADER__

Typedefs

- typedef struct grid2d grid2d
- typedef grid2d * pgrid2d
- typedef const grid2d * pcgrid2d

9.44.1 Detailed Description

Header file for FASP grid.

9.44.2 Macro Definition Documentation

9.44.2.1 __FASPGRID_HEADER__

```
#define __FASPGRID_HEADER__
```

indicate fasp_grid.h has been included before

Definition at line 12 of file fasp_grid.h.

9.44.3 Typedef Documentation

9.44.3.1 grid2d

```
typedef struct grid2d grid2d
```

2D grid type for plotting

9.44.3.2 pcgrid2d

```
typedef const grid2d* pcgrid2d
```

Grid in 2d

Definition at line 45 of file fasp_grid.h.

9.44.3.3 pgrid2d

```
typedef grid2d* pgrid2d
```

Grid in 2d

Definition at line 43 of file fasp_grid.h.

9.45 ItrSmootherBSR.c File Reference

Smoothers for dBSRmat matrices.

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

Functions

- void fasp_smoother_dbsr_jacobi (dBSRmat ∗A, dvector ∗b, dvector ∗u)

  Jacobi relaxation.
- void fasp_smoother_dbsr_jacobi_setup (dBSRmat ∗A, dvector ∗b, dvector ∗u, REAL ∗diaginv)

  Setup for jacobi relaxation, fetch the diagonal sub-block matrixes and make them inverse first.
- void fasp_smoother_dbsr_jacobi1 (dBSRmat ∗A, dvector ∗b, dvector ∗u, REAL ∗diaginv)

  Jacobi relaxation.
- void fasp_smoother_dbsr_gs (dBSRmat ∗A, dvector ∗b, dvector ∗u, INT order, INT ∗mark)

  Gauss-Seidel relaxation.
- void fasp_smoother_dbsr_gs1 (dBSRmat ∗A, dvector ∗b, dvector ∗u, INT order, INT ∗mark, REAL ∗diaginv)

  Gauss-Seidel relaxation.
- void fasp_smoother_dbsr_gs_ascend (dBSRmat ∗A, dvector ∗b, dvector ∗u, REAL ∗diaginv)

  Gauss-Seidel relaxation in the ascending order.
- void fasp_smoother_dbsr_gs_ascend1 (dBSRmat ∗A, dvector ∗b, dvector ∗u)

  Gauss-Seidel relaxation in the ascending order.
- void fasp_smoother_dbsr_gs_descend (dBSRmat ∗A, dvector ∗b, dvector ∗u, REAL ∗diaginv)

  Gauss-Seidel relaxation in the descending order.
- void fasp_smoother_dbsr_gs_descend1 (dBSRmat ∗A, dvector ∗b, dvector ∗u)

  Gauss-Seidel relaxation in the descending order.
- void fasp_smoother_dbsr_gs_order1 (dBSRmat ∗A, dvector ∗b, dvector ∗u, REAL ∗diaginv, INT ∗mark)

  Gauss-Seidel relaxation in the user-defined order.
- void fasp_smoother_dbsr_gs_order2 (dBSRmat ∗A, dvector ∗b, dvector ∗u, INT ∗mark, REAL ∗work)

  Gauss-Seidel relaxation in the user-defined order.
- void fasp_smoother_dbsr_sor (dBSRmat ∗A, dvector ∗b, dvector ∗u, INT order, INT ∗mark, REAL weight)

  SOR relaxation.
- void fasp_smoother_dbsr_sor1 (dBSRmat ∗A, dvector ∗b, dvector ∗u, INT order, INT ∗mark, REAL ∗diaginv, REAL weight)

  SOR relaxation.
- void fasp_smoother_dbsr_sor_ascend (dBSRmat ∗A, dvector ∗b, dvector ∗u, REAL ∗diaginv, REAL weight)

  SOR relaxation in the ascending order.
- void fasp_smoother_dbsr_sor_descend (dBSRmat ∗A, dvector ∗b, dvector ∗u, REAL ∗diaginv, REAL weight)

  SOR relaxation in the descending order.
- void fasp_smoother_dbsr_sor_order (dBSRmat ∗A, dvector ∗b, dvector ∗u, REAL ∗diaginv, INT ∗mark, REAL weight)

  SOR relaxation in the user-defined order.
- void fasp_smoother_dbsr_ilu (dBSRmat ∗A, dvector ∗b, dvector ∗x, void ∗data)

  ILU method as the smoother in solving Au=b with multigrid method.

Variables

- REAL ilu_solve_omp = 0.0

9.45.1 Detailed Description

Smoothers for dBSRmat matrices.

Note

This file contains Level-2 (Itr) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, AuxThreads.c, AuxTiming.c, BlaSmallMatInv.c, BlaSmallMat.c, BlaArray.c, BlaSpmvBSR.c, and PreBSR.c

// TODO: Need to optimize routines here! –Chensong

9.45.2 Function Documentation

9.45.2.1 fasp_smoother_dbsr_gs()

```
void fasp_smoother_dbsr_gs (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            INT order,
            INT * mark )
```

Gauss-Seidel relaxation.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
order	Flag to indicate the order for smoothing If mark = NULL ASCEND 12: in ascending order DESCEND 21: in descending order If mark != NULL: in the user-defined order
mark	Pointer to NULL or to the user-defined ordering

Author

Zhiyang Zhou

Date

2010/10/25

Modified by Chunsheng Feng, Zheng Li on 08/03/2012

Definition at line 426 of file ItrSmootherBSR.c.

9.45.2.2 fasp_smoother_dbsr_gs1()

```
void fasp_smoother_dbsr_gs1 (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            INT order,
            INT * mark,
            REAL * diaginv )
```

Gauss-Seidel relaxation.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
order	Flag to indicate the order for smoothing If mark = NULL ASCEND 12: in ascending order DESCEND 21: in descending order If mark != NULL: in the user-defined order
mark	Pointer to NULL or to the user-defined ordering
diaginv	Inverses for all the diagonal blocks of A

Author

Zhiyang Zhou

Date

2010/10/25

Definition at line 542 of file ItrSmootherBSR.c.

9.45.2.3 fasp_smoother_dbsr_gs_ascend()

```
void fasp_smoother_dbsr_gs_ascend (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv )
```

Gauss-Seidel relaxation in the ascending order.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial guess, OUT: approximation)
diaginv	Inverses for all the diagonal blocks of A

Author

Zhiyang Zhou

Date

2010/10/25

Definition at line 579 of file ItrSmootherBSR.c.

9.45.2.4 fasp_smoother_dbsr_gs_ascend1()

```
void fasp_smoother_dbsr_gs_ascend1 (
            dBSRmat * A,
            dvector * b,
            dvector * u )
```

Gauss-Seidel relaxation in the ascending order.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial guess, OUT: approximation)

Author

Xiaozhe 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 652 of file ItrSmootherBSR.c.

9.45.2.5 fasp_smoother_dbsr_gs_descend()

```
void fasp_smoother_dbsr_gs_descend (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv )
```

Gauss-Seidel relaxation in the descending order.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial guess, OUT: approximation)
diaginv	Inverses for all the diagonal blocks of A

Author

Zhiyang Zhou

Date

2010/10/25

Definition at line 721 of file ItrSmootherBSR.c.

9.45.2.6 fasp_smoother_dbsr_gs_descend1()

```
void fasp_smoother_dbsr_gs_descend1 (
            dBSRmat * A,
            dvector * b,
            dvector * u )
```

Gauss-Seidel relaxation in the descending order.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial guess, OUT: approximation)

Author

Xiaozhe Hu

Date

01/01/2014

Note

The only difference between the functions 'fasp_smoother_dbsr_gs_ascend1' and 'fasp_smoother_dbsr_gs_↩ascend' is that we don't have to multiply by the inverses of the diagonal blocks in each ROW since matrix A has been such scaled that all the diagonal blocks become identity matrices.

Definition at line 795 of file ItrSmootherBSR.c.

9.45.2.7 fasp_smoother_dbsr_gs_order1()

```
void fasp_smoother_dbsr_gs_order1 (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv,
            INT * mark )
```

Gauss-Seidel relaxation in the user-defined order.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial guess, OUT: approximation)
diaginv	Inverses for all the diagonal blocks of A
mark	Pointer to the user-defined ordering

Author

Zhiyang Zhou

Date

2010/10/25

Definition at line 865 of file ItrSmootherBSR.c.

9.45.2.8 fasp_smoother_dbsr_gs_order2()

```
void fasp_smoother_dbsr_gs_order2 (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            INT * mark,
            REAL * work )
```

Gauss-Seidel relaxation in the user-defined order.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial guess, OUT: approximation)
mark	Pointer to the user-defined ordering
work	Work temp array

Author

Zhiyang Zhou

Date

2010/11/08

Note

The only difference between the functions 'fasp_smoother_dbsr_gs_order2' and 'fasp_smoother_dbsr_gs_order1' lies in that we don't have to multiply by the inverses of the diagonal blocks in each ROW since matrix A has been such scaled that all the diagonal blocks become identity matrices.

Definition at line 943 of file ItrSmootherBSR.c.

9.45.2.9 fasp_smoother_dbsr_ilu()

```
void fasp_smoother_dbsr_ilu (
            dBSRmat * A,
            dvector * b,
            dvector * x,
            void * data )
```

ILU method as the smoother in solving Au=b with multigrid method.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
data	Pointer to user defined data

Author

Zhiyang Zhou, Zheng Li

Date

2010/10/25

NOTE: Add multi-threads parallel ILU block by Zheng Li 12/04/2016. form residual zr = b - A x

solve LU z=zr

x=x+z

Definition at line 1558 of file ItrSmootherBSR.c.

9.45.2.10 fasp_smoother_dbsr_jacobi()

```
void fasp_smoother_dbsr_jacobi (
            dBSRmat * A,
            dvector * b,
            dvector * u )
```

Jacobi relaxation.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)

Author

Zhiyang Zhou

Date

2010/10/25

Modified by Chunsheng Feng, Zheng Li on 08/02/2012

Definition at line 53 of file ItrSmootherBSR.c.

9.45.2.11 fasp_smoother_dbsr_jacobi1()

```
void fasp_smoother_dbsr_jacobi1 (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv )
```

Jacobi relaxation.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
diaginv	Inverses for all the diagonal blocks of A

Author

Zhiyang Zhou

Date

2010/10/25

Modified by Chunsheng Feng, Zheng Li on 08/03/2012

Definition at line 272 of file ItrSmootherBSR.c.

9.45.2.12 fasp_smoother_dbsr_jacobi_setup()

```
void fasp_smoother_dbsr_jacobi_setup (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv )
```

Setup for jacobi relaxation, fetch the diagonal sub-block matrixes and make them inverse first.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
diaginv	Inverse of the diagonal entries

Author

Zhiyang Zhou

Date

10/25/2010

Modified by Chunsheng Feng, Zheng Li on 08/02/2012

Definition at line 164 of file ItrSmootherBSR.c.

9.45.2.13 fasp_smoother_dbsr_sor()

```
void fasp_smoother_dbsr_sor (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            INT order,
            INT * mark,
            REAL weight )
```

SOR relaxation.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial guess, OUT: approximation)
order	Flag to indicate the order for smoothing If mark = NULL ASCEND 12: in ascending order DESCEND 21: in descending order If mark != NULL: in the user-defined order
mark	Pointer to NULL or to the user-defined ordering
weight	Over-relaxation weight

Author

Zhiyang Zhou

Date

2010/10/25

Modified by Chunsheng Feng, Zheng Li on 08/03/2012

Definition at line 1020 of file ItrSmootherBSR.c.

9.45.2.14 fasp_smoother_dbsr_sor1()

```
void fasp_smoother_dbsr_sor1 (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            INT order,
            INT * mark,
            REAL * diaginv,
            REAL weight )
```

SOR relaxation.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial guess, OUT: approximation)
order	Flag to indicate the order for smoothing If mark = NULL ASCEND 12: in ascending order DESCEND 21: in descending order If mark != NULL: in the user-defined order
mark	Pointer to NULL or to the user-defined ordering
diaginv	Inverses for all the diagonal blocks of A
weight	Over-relaxation weight

Author

Zhiyang Zhou

Date

2010/10/25

Definition at line 1142 of file ItrSmootherBSR.c.

9.45.2.15 fasp_smoother_dbsr_sor_ascend()

```
void fasp_smoother_dbsr_sor_ascend (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv,
            REAL weight )
```

SOR relaxation in the ascending order.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial guess, OUT: approximation)
diaginv	Inverses for all the diagonal blocks of A
weight	Over-relaxation weight

Author

Zhiyang Zhou

Date

2010/10/25

Modified by Chunsheng Feng, Zheng Li on 2012/09/04

Definition at line 1183 of file ItrSmootherBSR.c.

9.45.2.16 fasp_smoother_dbsr_sor_descend()

```
void fasp_smoother_dbsr_sor_descend (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv,
            REAL weight )
```

SOR relaxation in the descending order.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial guess, OUT: approximation)
diaginv	Inverses for all the diagonal blocks of A
weight	Over-relaxation weight

Author

Zhiyang Zhou

Date

2010/10/25

Modified by Chunsheng Feng, Zheng Li on 2012/09/04

Definition at line 1306 of file ItrSmootherBSR.c.

9.45.2.17 fasp_smoother_dbsr_sor_order()

```
void fasp_smoother_dbsr_sor_order (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv,
            INT * mark,
            REAL weight )
```

SOR relaxation in the user-defined order.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
diaginv	Inverses for all the diagonal blocks of A
mark	Pointer to the user-defined ordering
weight	Over-relaxation weight

Author

Zhiyang Zhou

Date

2010/10/25

Modified by Chunsheng Feng, Zheng Li on 2012/09/04

Definition at line 1432 of file ItrSmootherBSR.c.

9.45.3 Variable Documentation

9.45.3.1 ilu_solve_omp

```
REAL ilu_solve_omp = 0.0
```

ILU time for the SOLVE phase

Definition at line 33 of file ItrSmootherBSR.c.

9.46 ItrSmootherCSR.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)

  Weighted Jacobi method as a smoother.
- void fasp_smoother_dcsr_gs (dvector ∗u, const INT i_1, const INT i_n, const INT s, dCSRmat ∗A, dvector ∗b, INT L)

  Gauss-Seidel method as a smoother.
- void fasp_smoother_dcsr_gs_cf (dvector ∗u, dCSRmat ∗A, dvector ∗b, INT L, INT ∗mark, const INT order)

  Gauss-Seidel smoother with C/F ordering for Au=b.
- void fasp_smoother_dcsr_sgs (dvector ∗u, dCSRmat ∗A, dvector ∗b, INT L)

  Symmetric Gauss-Seidel method as a smoother.
- void fasp_smoother_dcsr_sor (dvector ∗u, const INT i_1, const INT i_n, const INT s, dCSRmat ∗A, dvector ∗b, INT L, const REAL w)

  SOR method as a smoother.
- void fasp_smoother_dcsr_sor_cf (dvector ∗u, dCSRmat ∗A, dvector ∗b, INT L, const REAL w, INT ∗mark, const INT order)

  SOR smoother with C/F ordering for Au=b.
- void fasp_smoother_dcsr_ilu (dCSRmat ∗A, dvector ∗b, dvector ∗x, void ∗data)

  ILU method as a smoother.
- void fasp_smoother_dcsr_kaczmarz (dvector ∗u, const INT i_1, const INT i_n, const INT s, dCSRmat ∗A, dvector ∗b, INT L, const REAL w)

  Kaczmarz method as a smoother.
- void fasp_smoother_dcsr_L1diag (dvector ∗u, const INT i_1, const INT i_n, const INT s, dCSRmat ∗A, dvector ∗b, INT L)

  Diagonal scaling (using L1 norm) as a smoother.
- void fasp_smoother_dcsr_gs_rb3d (dvector ∗u, dCSRmat ∗A, dvector ∗b, INT L, const INT order, INT ∗mark, const INT maximap, const INT nx, const INT ny, const INT nz)

  Colored Gauss-Seidel smoother for Au=b.

9.46.1 Detailed Description

Smoothers for dCSRmat matrices.

Note

This file contains Level-2 (Itr) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, AuxThreads.c, BlaArray.c, and BlaSpmvCSR.c

// TODO: Need to optimize subroutines here! –Chensong // TODO: Why should weight be fixed in wJacobi? –Chensong

9.46.2 Function Documentation

9.46.2.1 fasp_smoother_dcsr_gs()

```
void fasp_smoother_dcsr_gs (
            dvector * u,
            const INT i_1,
            const INT i_n,
            const INT s,
            dCSRmat * A,
            dvector * b,
            INT L )
```

Gauss-Seidel method as a smoother.

Parameters

u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
i←_←1	Starting index
i←_←n	Ending index
s	Increasing step
A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
L	Number of iterations

Author

Xuehai Huang, Chensong Zhang

Date

09/26/2009

Modified by Chunsheng Feng, Zheng Li on 09/01/2012

Definition at line 206 of file ItrSmootherCSR.c.

9.46.2.2 fasp_smoother_dcsr_gs_cf()

```
void fasp_smoother_dcsr_gs_cf (
            dvector * u,
            dCSRmat * A,
            dvector * b,
            INT L,
            INT * mark,
            const INT order )
```

Gauss-Seidel smoother with C/F ordering for Au=b.

Parameters

u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
L	Number of iterations
mark	C/F marker array
order	C/F ordering: -1: F-first; 1: C-first

Author

Zhiyang Zhou

Date

11/12/2010

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

Definition at line 375 of file ItrSmootherCSR.c.

9.46.2.3 fasp_smoother_dcsr_gs_rb3d()

```
void fasp_smoother_dcsr_gs_rb3d (
            dvector * u,
            dCSRmat * A,
            dvector * b,
            INT L,
            const INT order,
            INT * mark,
            const INT maximap,
            const INT nx,
            const INT ny,
            const INT nz )
```

Colored Gauss-Seidel smoother for Au=b.

Parameters

u	Initial guess and the new approximation to the solution
A	Pointer to stiffness matrix
b	Pointer to right hand side
L	Number of iterations
order	Ordering: -1: Forward; 1: Backward
mark	Marker for C/F points
maximap	Size of IMAP
nx	Number vertex of X direction
ny	Number vertex of Y direction
nz	Number vertex of Z direction

Author

Chunsheng Feng

Date

02/08/2012

Definition at line 1438 of file ItrSmootherCSR.c.

9.46.2.4 fasp_smoother_dcsr_ilu()

```
void fasp_smoother_dcsr_ilu (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            void * data )
```

ILU method as a smoother.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
x	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
data	Pointer to user defined data

Author

Shiquan Zhang, Xiaozhe Hu

Date

2010/11/12

form residual zr = b - A x

Definition at line 1079 of file ItrSmootherCSR.c.

9.46.2.5 fasp_smoother_dcsr_jacobi()

```
void fasp_smoother_dcsr_jacobi (
            dvector * u,
            const INT i_1,
            const INT i_n,
            const INT s,
            dCSRmat * A,
            dvector * b,
            INT L )
```

Weighted 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 71 of file ItrSmootherCSR.c.

9.46.2.6 fasp_smoother_dcsr_kaczmarz()

```
void fasp_smoother_dcsr_kaczmarz (
            dvector * u,
            const INT i_1,
            const INT i_n,
            const INT s,
            dCSRmat * A,
            dvector * b,
            INT L,
            const REAL w )
```

Kaczmarz method as a smoother.

Parameters

u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
i_1	Starting index
i_n	Ending index
s	Increasing step
A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
L	Number of iterations
w	Over-relaxation weight

Author

Xiaozhe Hu

Date

2010/11/12

Modified by Chunsheng Feng, Zheng Li on 2012/09/01

Definition at line 1158 of file ItrSmootherCSR.c.

9.46.2.7 fasp_smoother_dcsr_L1diag()

```
void fasp_smoother_dcsr_L1diag (
            dvector * u,
            const INT i_1,
            const INT i_n,
            const INT s,
            dCSRmat * A,
            dvector * b,
            INT L )
```

Diagonal scaling (using L1 norm) as a smoother.

Parameters

u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
i_1	Starting index
i_n	Ending index
s	Increasing step
A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
L	Number of iterations

Author

Xiaozhe Hu, James Brannick

Date

01/26/2011

Modified by Chunsheng Feng, Zheng Li on 09/01/2012

Definition at line 1299 of file ItrSmootherCSR.c.

9.46.2.8 fasp_smoother_dcsr_sgs()

```
void fasp_smoother_dcsr_sgs (
            dvector * u,
            dCSRmat * A,
            dvector * b,
            INT L )
```

Symmetric Gauss-Seidel method as a smoother.

Parameters

u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
L	Number of iterations

Author

Xiaozhe Hu

Date

10/26/2010

Modified by Chunsheng Feng, Zheng Li on 09/01/2012

Definition at line 640 of file ItrSmootherCSR.c.

9.46.2.9 fasp_smoother_dcsr_sor()

```
void fasp_smoother_dcsr_sor (
            dvector * u,
            const INT i_1,
            const INT i_n,
            const INT s,
            dCSRmat * A,
            dvector * b,
            INT L,
            const REAL w )
```

SOR method as a smoother.

Parameters

u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)

Parameters

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

10/26/2010

Modified by Chunsheng Feng, Zheng Li on 09/01/2012

Definition at line 757 of file ItrSmootherCSR.c.

9.46.2.10 fasp_smoother_dcsr_sor_cf()

```
void fasp_smoother_dcsr_sor_cf (
            dvector * u,
            dCSRmat * A,
            dvector * b,
            INT L,
            const REAL w,
            INT * mark,
            const INT order )
```

SOR smoother with C/F ordering for Au=b.

Parameters

u	Pointer to dvector: the unknowns (IN: initial, OUT: approximation)
A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
L	Number of iterations
w	Over-relaxation weight
mark	C/F marker array
order	C/F ordering: -1: F-first; 1: C-first

Author

Zhiyang Zhou

Date

2010/11/12

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

Definition at line 885 of file ItrSmootherCSR.c.

9.47 ItrSmootherCSRcr.c File Reference

Smoothers for dCSRmat matrices using compatible relaxation.

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

Functions

- void fasp_smoother_dcsr_gscr (INT pt, INT n, REAL ∗u, INT ∗ia, INT ∗ja, REAL ∗a, REAL ∗b, INT L, INT ∗CF)

  Gauss Seidel method restriced to a block.

9.47.1 Detailed Description

Smoothers for dCSRmat matrices using compatible relaxation.

Note

Restricted smoothers for compatible relaxation, C/F smoothing, etc.
This file contains Level-2 (Itr) functions. It requires: AuxMessage.c

// TODO: Need to optimize routines here! –Chensong

9.47.2 Function Documentation

9.47.2.1 fasp_smoother_dcsr_gscr()

```
void fasp_smoother_dcsr_gscr (
            INT pt,
            INT n,
            REAL * u,
            INT * ia,
            INT * ja,
            REAL * a,
            REAL * b,
            INT L,
            INT * CF )
```

Gauss Seidel method restriced to a block.

Parameters

pt	Relax type, e.g., cpt, fpt, etc..
n	Number of variables
u	Iterated solution
ia	Row pointer
ja	Column index
a	Pointers to sparse matrix values in CSR format
b	Pointer to right hand side
L	Number of iterations
CF	Marker for C, F points

Author

James Brannick

Date

09/07/2010

Note

Gauss Seidel CR smoother (Smoother_Type = 99)

Definition at line 48 of file ItrSmootherCSRcr.c.

9.48 ItrSmootherCSRpoly.c File Reference

Smoothers for dCSRmat matrices using poly. approx. to A^{-1}.

```
#include <math.h>
#include <time.h>
#include <float.h>
#include <limits.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "ItrAuxiliary.inl"
```

Functions

- void fasp_smoother_dcsr_poly (dCSRmat ∗Amat, dvector ∗brhs, dvector ∗usol, INT n, INT ndeg, INT L)

  poly approx to A^{-1} as MG smoother
- void fasp_smoother_dcsr_poly_old (dCSRmat ∗Amat, dvector ∗brhs, dvector ∗usol, INT n, INT ndeg, INT L)

  poly approx to A^{-1} as MG smoother: JK&LTZ2010

9.48.1 Detailed Description

Smoothers for dCSRmat matrices using poly. approx. to A^{-1}.

Note

This file contains Level-2 (Itr) functions. It requires: AuxArray.c, AuxMemory.c, AuxThreads.c, BlaArray.c, and BlaSpmvCSR.c

Reference: Johannes K. Kraus, Panayot S. Vassilevski, Ludmil T. Zikatanov Polynomial of best uniform approximation to x^{-1} and smoothing in two-leve methods, 2013.

Warning

Do NOT use auto-indentation in this file!

// TODO: Need to optimize routines here! –Chensong

9.48.2 Function Documentation

9.48.2.1 fasp_smoother_dcsr_poly()

```
void fasp_smoother_dcsr_poly (
            dCSRmat * Amat,
            dvector * brhs,
            dvector * usol,
            INT n,
            INT ndeg,
            INT L )
```

poly approx to A^{-1} as MG smoother

Parameters

Amat	Pointer to stiffness matrix, consider square matrix.
brhs	Pointer to right hand side
usol	Pointer to solution
n	Problem size
ndeg	Degree of poly
L	Number of iterations

Author

Fei Cao, Xiaozhe Hu

Date

05/24/2012

Definition at line 67 of file ItrSmootherCSRpoly.c.

9.48.2.2 fasp_smoother_dcsr_poly_old()

```
void fasp_smoother_dcsr_poly_old (
            dCSRmat * Amat,
            dvector * brhs,
            dvector * usol,
            INT n,
            INT ndeg,
            INT L )
```

poly approx to A^{-1} as MG smoother: JK&LTZ2010

Parameters

Amat	Pointer to stiffness matrix
brhs	Pointer to right hand side
usol	Pointer to solution
n	Problem size
ndeg	Degree of poly
L	Number of iterations

Author

James Brannick and Ludmil T Zikatanov

Date

06/28/2010

Modified by Chunsheng Feng, Zheng Li on 10/18/2012

Definition at line 167 of file ItrSmootherCSRpoly.c.

9.49 ItrSmootherSTR.c File Reference

Smoothers for dSTRmat matrices.

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

Functions

- void fasp_smoother_dstr_jacobi (dSTRmat *A, dvector *b, dvector *u)

  Jacobi method as the smoother.
- void fasp_smoother_dstr_jacobi1 (dSTRmat *A, dvector *b, dvector *u, REAL *diaginv)

  Jacobi method as the smoother with diag_inv given.
- void fasp_smoother_dstr_gs (dSTRmat *A, dvector *b, dvector *u, const INT order, INT *mark)

  Gauss-Seidel method as the smoother.
- void fasp_smoother_dstr_gs1 (dSTRmat *A, dvector *b, dvector *u, const INT order, INT *mark, REAL *diaginv)

  Gauss-Seidel method as the smoother with diag_inv given.
- void fasp_smoother_dstr_gs_ascend (dSTRmat *A, dvector *b, dvector *u, REAL *diaginv)

  Gauss-Seidel method as the smoother in the ascending manner.
- void fasp_smoother_dstr_gs_descend (dSTRmat *A, dvector *b, dvector *u, REAL *diaginv)

  Gauss-Seidel method as the smoother in the descending manner.
- void fasp_smoother_dstr_gs_order (dSTRmat *A, dvector *b, dvector *u, REAL *diaginv, INT *mark)

  Gauss method as the smoother in the user-defined order.
- void fasp_smoother_dstr_gs_cf (dSTRmat *A, dvector *b, dvector *u, REAL *diaginv, INT *mark, const INT order)

  Gauss method as the smoother in the C-F manner.
- void fasp_smoother_dstr_sor (dSTRmat *A, dvector *b, dvector *u, const INT order, INT *mark, const REAL weight)

  SOR method as the smoother.
- void fasp_smoother_dstr_sor1 (dSTRmat *A, dvector *b, dvector *u, const INT order, INT *mark, REAL *diaginv, const REAL weight)

  SOR method as the smoother.
- void fasp_smoother_dstr_sor_ascend (dSTRmat *A, dvector *b, dvector *u, REAL *diaginv, REAL weight)

  SOR method as the smoother in the ascending manner.
- void fasp_smoother_dstr_sor_descend (dSTRmat *A, dvector *b, dvector *u, REAL *diaginv, REAL weight)

  SOR method as the smoother in the descending manner.

- void fasp_smoother_dstr_sor_order (dSTRmat *A, dvector *b, dvector *u, REAL *diaginv, INT *mark, REAL weight)

    SOR method as the smoother in the user-defined order.
- void fasp_smoother_dstr_sor_cf (dSTRmat *A, dvector *b, dvector *u, REAL *diaginv, INT *mark, const INT order, const REAL weight)

    SOR method as the smoother in the C-F manner.
- void fasp_generate_diaginv_block (dSTRmat *A, ivector *neigh, dvector *diaginv, ivector *pivot)

    Generate inverse of diagonal block for block smoothers.
- void fasp_smoother_dstr_swz (dSTRmat *A, dvector *b, dvector *u, dvector *diaginv, ivector *pivot, ivector *neigh, ivector *order)

    Schwarz method as the smoother.

9.49.1 Detailed Description

Smoothers for dSTRmat matrices.

Note

This file contains Level-2 (Itr) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, BlaSmallMat.c, BlaSmallMatInv.c, BlaSmallMatLU.c, and BlaSpmvSTR.c

9.49.2 Function Documentation

9.49.2.1 fasp_generate_diaginv_block()

```
void fasp_generate_diaginv_block (
            dSTRmat * A,
            ivector * neigh,
            dvector * diaginv,
            ivector * pivot )
```

Generate inverse of diagonal block for block smoothers.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
neigh	Pointer to ivector: neighborhoods
diaginv	Pointer to dvector: the inverse of the diagonals
pivot	Pointer to ivector: the pivot of diagonal blocks

Author

Xiaozhe Hu

Date

10/01/2011

Definition at line 1535 of file ItrSmootherSTR.c.

9.49.2.2 fasp_smoother_dstr_gs()

```
void fasp_smoother_dstr_gs (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            const INT order,
            INT * mark )
```

Gauss-Seidel method as the smoother.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
order	Flag to indicate the order for smoothing If mark = NULL ASCEND 12: in ascending manner DESCEND 21: in descending manner If mark != NULL USERDEFINED 0 : in the user-defined manner CPFIRST 1 : C-points first and then F-points FPFIRST -1 : F-points first and then C-points
mark	Pointer to the user-defined ordering(when order=0) or CF_marker array(when order!=0)

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 216 of file ItrSmootherSTR.c.

9.49.2.3 fasp_smoother_dstr_gs1()

```
void fasp_smoother_dstr_gs1 (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            const INT order,
            INT * mark,
            REAL * diaginv )
```

Gauss-Seidel method as the smoother with diag_inv given.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
order	Flag to indicate the order for smoothing If mark = NULL ASCEND 12: in ascending manner DESCEND 21: in descending manner If mark != NULL USERDEFINED 0 : in the user-defined manner CPFIRST 1 : C-points first and then F-points FPFIRST -1 : F-points first and then C-points
mark	Pointer to the user-defined ordering(when order=0) or CF_marker array(when order!=0)
diaginv	All the inverse matrices for all the diagonal block of A when (A->nc)>1, and NULL when (A->nc)=1

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 276 of file ItrSmootherSTR.c.

9.49.2.4 fasp_smoother_dstr_gs_ascend()

```
void fasp_smoother_dstr_gs_ascend (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv )
```

Gauss-Seidel method as the smoother in the ascending manner.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
diaginv	All the inverse matrices for all the diagonal block of A when (A->nc)>1, and NULL when (A->nc)=1

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 321 of file ItrSmootherSTR.c.

9.49.2.5 fasp_smoother_dstr_gs_cf()

```
void fasp_smoother_dstr_gs_cf (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv,
            INT * mark,
            const INT order )
```

Gauss method as the smoother in the C-F manner.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
diaginv	All the inverse matrices for all the diagonal block of A when (A->nc)>1, and NULL when (A->nc)=1
mark	Pointer to the user-defined order array
order	Flag to indicate the order for smoothing CPFIRST 1 : C-points first and then F-points FPFIRST -1 : F-points first and then C-points

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 676 of file ItrSmootherSTR.c.

9.49.2.6 fasp_smoother_dstr_gs_descend()

```
void fasp_smoother_dstr_gs_descend (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv )
```

Gauss-Seidel method as the smoother in the descending manner.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
diaginv	All the inverse matrices for all the diagonal block of A when (A->nc)>1, and NULL when (A->nc)=1

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 436 of file ItrSmootherSTR.c.

9.49.2.7 fasp_smoother_dstr_gs_order()

```
void fasp_smoother_dstr_gs_order (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv,
            INT * mark )
```

Gauss method as the smoother in the user-defined order.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
diaginv	All the inverse matrices for all the diagonal block of A when (A->nc)>1, and NULL when (A->nc)=1
mark	Pointer to the user-defined order array

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 553 of file ItrSmootherSTR.c.

9.49.2.8 fasp_smoother_dstr_jacobi()

```
void fasp_smoother_dstr_jacobi (
            dSTRmat * A,
            dvector * b,
            dvector * u )
```

Jacobi method as the smoother.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 43 of file ItrSmootherSTR.c.

9.49.2.9 fasp_smoother_dstr_jacobi1()

```
void fasp_smoother_dstr_jacobi1 (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv )
```

Jacobi method as the smoother with diag_inv given.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
diaginv	All the inverse matrices for all the diagonal block of A when (A->nc)>1, and NULL when (A->nc)=1

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 92 of file ItrSmootherSTR.c.

9.49.2.10 fasp_smoother_dstr_sor()

```
void fasp_smoother_dstr_sor (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            const INT order,
            INT * mark,
            const REAL weight )
```

SOR method as the smoother.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
order	Flag to indicate the order for smoothing If mark = NULL ASCEND 12: in ascending manner DESCEND 21: in descending manner If mark != NULL USERDEFINED 0 : in the user-defined manner CPFIRST 1 : C-points first and then F-points FPFIRST -1 : F-points first and then C-points
mark	Pointer to the user-defined ordering(when order=0) or CF_marker array(when order!=0)
weight	Over-relaxation weight

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 868 of file ItrSmootherSTR.c.

9.49.2.11 fasp_smoother_dstr_sor1()

```
void fasp_smoother_dstr_sor1 (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            const INT order,
            INT * mark,
            REAL * diaginv,
            const REAL weight )
```

SOR method as the smoother.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
order	Flag to indicate the order for smoothing If mark = NULL ASCEND 12: in ascending manner DESCEND 21: in descending manner If mark != NULL USERDEFINED 0 : in the user-defined manner CPFIRST 1 : C-points first and then F-points FPFIRST -1 : F-points first and then C-points
mark	Pointer to the user-defined ordering(when order=0) or CF_marker array(when order!=0)
diaginv	Inverse of the diagonal entries
weight	Over-relaxation weight

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 930 of file ItrSmootherSTR.c.

9.49.2.12 fasp_smoother_dstr_sor_ascend()

```
void fasp_smoother_dstr_sor_ascend (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv,
            REAL weight )
```

SOR method as the smoother in the ascending manner.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
diaginv	All the inverse matrices for all the diagonal block of A when (A->nc)>1, and NULL when (A->nc)=1
weight	Over-relaxation weight

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 976 of file ItrSmootherSTR.c.

9.49.2.13 fasp_smoother_dstr_sor_cf()

```
void fasp_smoother_dstr_sor_cf (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv,
            INT * mark,
            const INT order,
            const REAL weight )
```

SOR method as the smoother in the C-F manner.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
diaginv	All the inverse matrices for all the diagonal block of A when (A->nc)>1, and NULL when (A->nc)=1
mark	Pointer to the user-defined order array
order	Flag to indicate the order for smoothing CPFIRST 1 : C-points first and then F-points FPFIRST -1 : F-points first and then C-points

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 1348 of file ItrSmootherSTR.c.

9.49.2.14 fasp_smoother_dstr_sor_descend()

```
void fasp_smoother_dstr_sor_descend (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv,
            REAL weight )
```

SOR method as the smoother in the descending manner.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
diaginv	All the inverse matrices for all the diagonal block of A when (A->nc)>1, and NULL when (A->nc)=1
weight	Over-relaxation weight

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 1096 of file ItrSmootherSTR.c.

9.49.2.15 fasp_smoother_dstr_sor_order()

```
void fasp_smoother_dstr_sor_order (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            REAL * diaginv,
            INT * mark,
            REAL weight )
```

SOR method as the smoother in the user-defined order.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
diaginv	All the inverse matrices for all the diagonal block of A when (A->nc)>1, and NULL when (A->nc)=1
mark	Pointer to the user-defined order array
weight	Over-relaxation weight

Author

Shiquan Zhang, Zhiyang Zhou

Date

10/10/2010

Definition at line 1217 of file ItrSmootherSTR.c.

9.49.2.16 fasp_smoother_dstr_swz()

```
void fasp_smoother_dstr_swz (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            dvector * diaginv,
            ivector * pivot,
            ivector * neigh,
            ivector * order )
```

Schwarz method as the smoother.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
diaginv	Pointer to dvector: the inverse of the diagonals
pivot	Pointer to ivector: the pivot of diagonal blocks
neigh	Pointer to ivector: neighborhoods
order	Pointer to ivector: the smoothing order

Author

Xiaozhe Hu

Date

10/01/2011

Definition at line 1657 of file ItrSmootherSTR.c.

9.50 KryPbcgs.c File Reference

Krylov subspace methods – Preconditioned BiCGstab.

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

Functions

- INT fasp_solver_dcsr_pbcgs (dCSRmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned BiCGstab method for solving Au=b.
- INT fasp_solver_dbsr_pbcgs (dBSRmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned BiCGstab method for solving Au=b.
- INT fasp_solver_dblc_pbcgs (dBLCmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  A preconditioned BiCGstab method for solving Au=b.
- INT fasp_solver_dstr_pbcgs (dSTRmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned BiCGstab method for solving Au=b.
- INT fasp_solver_pbcgs (mxv_matfree ∗mf, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned BiCGstab method for solving Au=b.

9.50.1 Detailed Description

Krylov subspace methods – Preconditioned BiCGstab.

Note

This file contains Level-3 (Kry) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, BlaArray.c, BlaSpmvBLC.c, BlaSpmvBSR.c, BlaSpmvCSR.c, and BlaSpmvSTR.c
See KrySPbcgs.c for a safer version

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

TODO: Use one single function for all! –Chensong

9.50.2 Function Documentation

9.50.2.1 fasp_solver_dblc_pbcgs()

```
INT fasp_solver_dblc_pbcgs (
            dBLCmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

A preconditioned BiCGstab method for solving Au=b.

Parameters

A	Pointer to coefficient matrix
b	Pointer to dvector of right hand side
u	Pointer to dvector of DOFs
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

05/24/2010

Rewritten by Chensong Zhang on 04/30/2012

Definition at line 744 of file KryPbcgs.c.

9.50.2.2 fasp_solver_dbsr_pbcgs()

```
INT fasp_solver_dbsr_pbcgs (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned BiCGstab method for solving Au=b.

Parameters

A	Pointer to coefficient matrix
b	Pointer to dvector of right hand side
u	Pointer to dvector of DOFs
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

09/09/2009

Modified by Feiteng Huang on 06/01/2012: fix restart param-init Modified by Chensong Zhang on 03/31/2013

Definition at line 404 of file KryPbcgs.c.

9.50.2.3 fasp_solver_dcsr_pbcgs()

```
INT fasp_solver_dcsr_pbcgs (
            dCSRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned BiCGstab method for solving Au=b.

Parameters

A	Pointer to coefficient matrix
b	Pointer to dvector of right hand side
u	Pointer to dvector of DOFs
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

09/09/2009

Modified by Feiteng Huang on 06/01/2012: fix restart param-init Modified by Chensong Zhang on 03/31/2013

Definition at line 63 of file KryPbcgs.c.

9.50.2.4 fasp_solver_dstr_pbcgs()

```
INT fasp_solver_dstr_pbcgs (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned BiCGstab method for solving Au=b.

Parameters

A	Pointer to coefficient matrix
b	Pointer to dvector of right hand side
u	Pointer to dvector of DOFs
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou

Date

04/25/2010

Rewritten by Chensong Zhang on 04/30/2012

Definition at line 1084 of file KryPbcgs.c.

9.50.2.5 fasp_solver_pbcgs()

```
INT fasp_solver_pbcgs (
            mxv_matfree * mf,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned BiCGstab method for solving Au=b.

Parameters

mf	Pointer to mxv_matfree: spmv operation
b	Pointer to dvector: right hand side
u	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

09/09/2009

Rewritten by Chensong Zhang on 04/30/2012

Definition at line 1424 of file KryPbcgs.c.

9.51 KryPcg.c File Reference

Krylov subspace methods – Preconditioned CG.

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

Functions

- INT fasp_solver_dcsr_pcg (dCSRmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned conjugate gradient method for solving Au=b.
- INT fasp_solver_dbsr_pcg (dBSRmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned conjugate gradient method for solving Au=b.
- INT fasp_solver_dblc_pcg (dBLCmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned conjugate gradient method for solving Au=b.
- INT fasp_solver_dstr_pcg (dSTRmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

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

  Preconditioned conjugate gradient (CG) method for solving Au=b.

9.51.1 Detailed Description

Krylov subspace methods – Preconditioned CG.

Note

This file contains Level-3 (Kry) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, BlaArray.c, Bla↩ SpmvBLC.c, BlaSpmvBSR.c, BlaSpmvCSR.c, and BlaSpmvSTR.c
See KrySPcg.c for a safer version

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

TODO: Use one single function for all! –Chensong

Abstract algorithm

PCG method to solve A∗x=b is to generate {x_k} to approximate x

Step 0. Given A, b, x_0, M

Step 1. Compute residual r_0 = b-A∗x_0 and convergence check;

Step 2. Initialization z_0 = M^{-1}∗r_0, p_0=z_0;

Step 3. Main loop ...

FOR k = 0:MaxIt

- get step size alpha = f(r_k,z_k,p_k);
- update solution: x_{k+1} = x_k + alpha∗p_k;
- perform stagnation check;
- update residual: r_{k+1} = r_k - alpha∗(A∗p_k);
- perform residual check;
- obtain p_{k+1} using {p_0, p_1, ... , p_k};
- prepare for next iteration;
- print the result of k-th iteration; END FOR

Convergence check: norm(r)/norm(b) < tol

Stagnation check:

- IF norm(alpha∗p_k)/norm(x_{k+1}) < tol_stag
  1. compute r=b-A∗x_{k+1};
  2. convergence check;
  3. IF (not converged & restart_number < Max_Stag_Check) restart;
- END IF

Residual check:

- IF norm(r_{k+1})/norm(b) < tol
  1. compute the real residual r = b-A∗x_{k+1};
  2. convergence check;
  3. IF (not converged & restart_number < Max_Res_Check) restart;
- END IF

9.51.2 Function Documentation

9.51.2.1 fasp_solver_dblc_pcg()

```
INT fasp_solver_dblc_pcg (
            dBLCmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned conjugate gradient method for solving Au=b.

Parameters

A	Pointer to dBLCmat: coefficient matrix
b	Pointer to dvector: right hand side
u	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

05/24/2010

Modified by Chensong Zhang on 03/28/2013

Definition at line 678 of file KryPcg.c.

9.51.2.2 fasp_solver_dbsr_pcg()

```
INT fasp_solver_dbsr_pcg (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned conjugate gradient method for solving Au=b.

Parameters

A	Pointer to dBSRmat: coefficient matrix
b	Pointer to dvector: right hand side
u	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

05/26/2014

Definition at line 387 of file KryPcg.c.

9.51.2.3 fasp_solver_dcsr_pcg()

```
INT fasp_solver_dcsr_pcg (
            dCSRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned conjugate gradient method for solving Au=b.

Parameters

A	Pointer to dCSRmat: coefficient matrix
b	Pointer to dvector: right hand side
u	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang, Xiaozhe Hu, Shiquan Zhang

Date

05/06/2010

Definition at line 98 of file KryPcg.c.

9.51.2.4 fasp_solver_dstr_pcg()

```
INT fasp_solver_dstr_pcg (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned conjugate gradient method for solving Au=b.

Parameters

A	Pointer to dSTRmat: coefficient matrix
b	Pointer to dvector: right hand side
u	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou

Date

04/25/2010

Modified by Chensong Zhang on 03/28/2013

Definition at line 969 of file KryPcg.c.

9.51.2.5 fasp_solver_pcg()

```
INT fasp_solver_pcg (
            mxv_matfree * mf,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned conjugate gradient (CG) method for solving Au=b.

Parameters

mf	Pointer to mxv_matfree: spmv operation
b	Pointer to dvector: right hand side
u	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang, Xiaozhe Hu, Shiquan Zhang

Date

05/06/2010

Modified by Feiteng Huang on 09/19/2012: matrix free

Definition at line 1260 of file KryPcg.c.

9.52 KryPgcg.c File Reference

Krylov subspace methods – Preconditioned generalized CG.

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

Functions

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

  Preconditioned generilzed conjugate gradient (GCG) method for solving Au=b.
- INT fasp_solver_pgcg (mxv_matfree *mf, dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

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

9.52.1 Detailed Description

Krylov subspace methods – Preconditioned generalized CG.

Note

This file contains Level-3 (Kry) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, BlaArray.c, and BlaSpmvCSR.c

Reference: 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

TODO: Use one single function for all! –Chensong

9.52.2 Function Documentation

9.52.2.1 fasp_solver_dcsr_pgcg()

```
INT fasp_solver_dcsr_pgcg (
            dCSRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

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

Parameters

A	Pointer to dCSRmat: coefficient matrix
b	Pointer to dvector: right hand side
u	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

01/01/2012

Modified by Chensong Zhang on 05/01/2012

Definition at line 60 of file KryPgcg.c.

9.52.2.2 fasp_solver_pgcg()

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

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

Parameters

mf	Pointer to mxv_matfree: spmv operation
b	Pointer to dvector: right hand side
u	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type – Not implemented

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

01/01/2012

Note

Not completely implemented yet! –Chensong

Modified by Feiteng Huang on 09/26/2012: matrix free

Definition at line 210 of file KryPgcg.c.

9.53 KryPgcr.c File Reference

Krylov subspace methods – Preconditioned GCR.

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

Functions

- INT fasp_solver_dcsr_pgcr (dCSRmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  A preconditioned GCR method for solving Au=b.

9.53.1 Detailed Description

Krylov subspace methods – Preconditioned GCR.

Note

This file contains Level-3 (Kry) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, BlaArray.c, Bla←SpmvCSR.c, and BlaVector.c

TODO: Use one single function for all! –Chensong

9.53.2 Function Documentation

9.53.2.1 fasp_solver_dcsr_pgcr()

```
INT fasp_solver_dcsr_pgcr (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

A preconditioned GCR method for solving Au=b.

Parameters

A	Pointer to coefficient matrix
b	Pointer to dvector of right hand side
x	Pointer to dvector of dofs
pc	Pointer to structure of precondition (precond)
tol	Tolerance for stopage
MaxIt	Maximal number of iterations
restart	Restart number for GCR
stop_type	Stopping type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Note

Refer to YVAN NOTAY "AN AGGREGATION-BASED ALGEBRAIC MULTIGRID METHOD"

Author

Zheng Li

Date

12/23/2014

Definition at line 53 of file KryPgcr.c.

9.54 KryPgmres.c File Reference

Krylov subspace methods – Right-preconditioned GMRes.

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

Functions

- INT fasp_solver_dcsr_pgmres (dCSRmat *A, dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Right preconditioned GMRES method for solving Au=b.
- INT fasp_solver_dblc_pgmres (dBLCmat *A, dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned GMRES method for solving Au=b.
- INT fasp_solver_dbsr_pgmres (dBSRmat *A, dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned GMRES method for solving Au=b.
- INT fasp_solver_dstr_pgmres (dSTRmat *A, dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned GMRES method for solving Au=b.
- INT fasp_solver_pgmres (mxv_matfree *mf, dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Solve "Ax=b" using PGMRES (right preconditioned) iterative method.

9.54.1 Detailed Description

Krylov subspace methods – Right-preconditioned GMRes.

Note

This file contains Level-3 (Kry) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, BlaArray.c, Bla↩
SpmvBLC.c, BlaSpmvBSR.c, BlaSpmvCSR.c, and BlaSpmvSTR.c
See also KryPvgmres.c for a variable restarting version.
See KrySPgmres.c for a safer version

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

TODO: Use one single function for all! –Chensong

9.54.2 Function Documentation

9.54.2.1 fasp_solver_dblc_pgmres()

```
INT fasp_solver_dblc_pgmres (
            dBLCmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned GMRES method for solving Au=b.

Parameters

A	Pointer to dBLCmat: coefficient matrix
b	Pointer to dvector: right hand side
x	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

05/24/2010

Modified by Chensong Zhang on 04/05/2013: add stop_type and safe check

Definition at line 369 of file KryPgmres.c.

9.54.2.2 fasp_solver_dbsr_pgmres()

```
INT fasp_solver_dbsr_pgmres (
            dBSRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned GMRES method for solving Au=b.

Parameters

A	Pointer to dBSRmat: coefficient matrix
b	Pointer to dvector: right hand side
x	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou

Date

2010/12/21

Modified by Chensong Zhang on 04/05/2013: add stop_type and safe check

Definition at line 671 of file KryPgmres.c.

9.54.2.3 fasp_solver_dcsr_pgmres()

```
INT fasp_solver_dcsr_pgmres (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Right preconditioned GMRES method for solving Au=b.

Parameters

A	Pointer to dCSRmat: coefficient matrix
b	Pointer to dvector: right hand side
x	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou

Date

2010/11/28

Modified by Chensong Zhang on 04/05/2013: Add stop_type and safe check Modified by Chunsheng Feng on 07/22/2013: Add adapt memory allocate Modified by Chensong Zhang on 09/21/2014: Add comments and reorganize code

Definition at line 67 of file KryPgmres.c.

9.54.2.4 fasp_solver_dstr_pgmres()

```
INT fasp_solver_dstr_pgmres (
            dSTRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned GMRES method for solving Au=b.

Parameters

A	Pointer to dSTRmat: coefficient matrix
b	Pointer to dvector: right hand side
x	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou

Date

2010/11/28

Modified by Chensong Zhang on 04/05/2013: add stop_type and safe check

Definition at line 974 of file KryPgmres.c.

9.54.2.5 fasp_solver_pgmres()

```
INT fasp_solver_pgmres (
            mxv_matfree * mf,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Solve "Ax=b" using PGMRES (right preconditioned) iterative method.

Parameters

mf	Pointer to mxv_matfree: spmv operation
b	Pointer to dvector: right hand side
x	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type – DOES not support this parameter
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou

Date

2010/11/28

Modified by Chunsheng Feng on 07/22/2013: Add adapt memory allocate

Definition at line 1276 of file KryPgmres.c.

9.55 KryPminres.c File Reference

Krylov subspace methods – Preconditioned minimal residual.

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

Functions

- INT fasp_solver_dcsr_pminres (dCSRmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

    A preconditioned minimal residual (Minres) method for solving Au=b.
- INT fasp_solver_dblc_pminres (dBLCmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

    A preconditioned minimal residual (Minres) method for solving Au=b.
- INT fasp_solver_dstr_pminres (dSTRmat ∗A, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

    A preconditioned minimal residual (Minres) method for solving Au=b.
- INT fasp_solver_pminres (mxv_matfree ∗mf, dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

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

9.55.1 Detailed Description

Krylov subspace methods – Preconditioned minimal residual.

Note

This file contains Level-3 (Kry) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, BlaArray.c, Bla↩ SpmvBLC.c, BlaSpmvCSR.c, and BlaSpmvSTR.c.o
See KrySPminres.c for a safer version

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

TODO: Use one single function for all! –Chensong

9.55.2 Function Documentation

9.55.2.1 fasp_solver_dblc_pminres()

```
INT fasp_solver_dblc_pminres (
            dBLCmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

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

Parameters

A	Pointer to dBLCmat: coefficient matrix
b	Pointer to dvector: right hand side
u	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

05/01/2012

Rewritten based on the original version by Xiaozhe Hu 05/24/2010 Modified by Chensong Zhang on 04/09/2013

Definition at line 468 of file KryPminres.c.

9.55.2.2 fasp_solver_dcsr_pminres()

```
INT fasp_solver_dcsr_pminres (
            dCSRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

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

Parameters

A	Pointer to dCSRmat: coefficient matrix
b	Pointer to dvector: right hand side
u	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

05/01/2012

Rewritten based on the original version by Shiquan Zhang 05/10/2010 Modified by Chensong Zhang on 04/09/2013

Definition at line 62 of file KryPminres.c.

9.55.2.3 fasp_solver_dstr_pminres()

```
INT fasp_solver_dstr_pminres (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

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

Parameters

A	Pointer to dSTRmat: coefficient matrix
b	Pointer to dvector: right hand side
u	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/09/2013

Definition at line 871 of file KryPminres.c.

9.55.2.4 fasp_solver_pminres()

```
INT fasp_solver_pminres (
            mxv_matfree * mf,
            dvector * b,
            dvector * u,
```

```
precond * pc,
const REAL tol,
const INT MaxIt,
const SHORT stop_type,
const SHORT prtlvl )
```

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

Parameters

mf	Pointer to mxv_matfree: spmv operation
b	Pointer to dvector: right hand side
u	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Shiquan Zhang

Date

10/24/2010

Rewritten by Chensong Zhang on 05/01/2012

Definition at line 1276 of file KryPminres.c.

9.56 KryPvbcgs.c File Reference

Krylov subspace methods – Preconditioned BiCGstab.

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

Functions

- INT fasp_solver_dcsr_pvbcgs (dCSRmat *A, dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned BiCGstab method for solving Au=b, Rewritten from Matlab 2011a.
- INT fasp_solver_dbsr_pvbcgs (dBSRmat *A, dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned BiCGstab method for solving Au=b, Rewritten from Matlab 2011a.
- INT fasp_solver_dblc_pvbcgs (dBLCmat *A, dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned BiCGstab method for solving Au=b, Rewritten from Matlab 2011a.
- INT fasp_solver_dstr_pvbcgs (dSTRmat *A, dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned BiCGstab method for solving Au=b, Rewritten from Matlab 2011a.
- INT fasp_solver_pvbcgs (mxv_matfree *mf, dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT stop_type, const SHORT prtlvl)

  Preconditioned BiCGstab method for solving Au=b, Rewritten from Matlab 2011a.

9.56.1 Detailed Description

Krylov subspace methods – Preconditioned BiCGstab.

Note

> This file contains Level-3 (Kry) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, BlaArray.c, BlaSpmvBLC.c, BlaSpmvBSR.c, BlaSpmvCSR.c, and BlaSpmvSTR.c
> See KrySPbcgs.c for a safer version

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

TODO: Is this version better? Remove the older version! –Chensong TODO: Use one single function for all! –Chensong

9.56.2 Function Documentation

9.56.2.1 fasp_solver_dblc_pvbcgs()

```
INT fasp_solver_dblc_pvbcgs (
            dBLCmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned BiCGstab method for solving Au=b, Rewritten from Matlab 2011a.

Parameters

A	Pointer to coefficient matrix
b	Pointer to dvector of right hand side
u	Pointer to dvector of DOFs
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chunsheng Feng

Date

03/04/2016

Definition at line 724 of file KryPvbcgs.c.

9.56.2.2 fasp_solver_dbsr_pvbcgs()

```
INT fasp_solver_dbsr_pvbcgs (
            dBSRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned BiCGstab method for solving Au=b, Rewritten from Matlab 2011a.

Parameters

A	Pointer to coefficient matrix
b	Pointer to dvector of right hand side
u	Pointer to dvector of DOFs
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chunsheng Feng

Date

03/04/2016

Definition at line 393 of file KryPvbcgs.c.

9.56.2.3 fasp_solver_dcsr_pvbcgs()

```
INT fasp_solver_dcsr_pvbcgs (
            dCSRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned BiCGstab method for solving Au=b, Rewritten from Matlab 2011a.

Parameters

A	Pointer to coefficient matrix
b	Pointer to dvector of right hand side
u	Pointer to dvector of DOFs
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chunsheng Feng

Date

03/04/2016

Definition at line 62 of file KryPvbcgs.c.

9.56.2.4 fasp_solver_dstr_pvbcgs()

```
INT fasp_solver_dstr_pvbcgs (
            dSTRmat * A,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned BiCGstab method for solving Au=b, Rewritten from Matlab 2011a.

Parameters

A	Pointer to coefficient matrix
b	Pointer to dvector of right hand side
u	Pointer to dvector of DOFs
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chunsheng Feng

Date

03/04/2016

Definition at line 1055 of file KryPvbcgs.c.

9.56.2.5 fasp_solver_pvbcgs()

```
INT fasp_solver_pvbcgs (
            mxv_matfree * mf,
            dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Preconditioned BiCGstab method for solving Au=b, Rewritten from Matlab 2011a.

Parameters

mf	Pointer to mxv_matfree: spmv operation
b	Pointer to dvector of right hand side
u	Pointer to dvector of DOFs
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chunsheng Feng

Date

03/04/2016

Definition at line 1386 of file KryPvbcgs.c.

9.57 KryPvfgmres.c File Reference

Krylov subspace methods – Preconditioned variable-restarting FGMRes.

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

Functions

- INT fasp_solver_dcsr_pvfgmres (dCSRmat *A, dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Solve "Ax=b" using PFGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during iteration and flexible preconditioner can be used.
- INT fasp_solver_dbsr_pvfgmres (dBSRmat *A, dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Solve "Ax=b" using PFGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during iteration and flexible preconditioner can be used.
- INT fasp_solver_dblc_pvfgmres (dBLCmat *A, dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Solve "Ax=b" using PFGMRES (right preconditioned) iterative method in which the restart parameter can be adaptively modified during iteration and flexible preconditioner can be used.
- INT fasp_solver_pvfgmres (mxv_matfree *mf, dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Solve "Ax=b" using PFGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during iteration and flexible preconditioner can be used.

9.57.1 Detailed Description

Krylov subspace methods – Preconditioned variable-restarting FGMRes.

Note

This file contains Level-3 (Kry) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, BlaArray.c, BlaSpmvBLC.c, BlaSpmvBSR.c, and BlaSpmvCSR.c
This file is modifed from KryPvgmres.c

Reference: 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.

TODO: Use one single function for all! –Chensong

9.57.2 Function Documentation

9.57.2.1 fasp_solver_dblc_pvfgmres()

```
INT fasp_solver_dblc_pvfgmres (
            dBLCmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Solve "Ax=b" using PFGMRES (right preconditioned) iterative method in which the restart parameter can be adaptively modified during iteration and flexible preconditioner can be used.

Parameters

A	Pointer to coefficient matrix
b	Pointer to right hand side vector
x	Pointer to solution vector
MaxIt	Maximal iteration number allowed
tol	Tolerance
pc	Pointer to preconditioner data
prtlvl	How much information to print out
stop_type	Stopping criterion, i.e.||r_k||/||r_0||<tol
restart	Number of restart for GMRES

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

01/04/2012

Note

Based on Zhiyang Zhou's pvgmres.c

Modified by Chunsheng Feng on 07/22/2013: Add adaptive memory allocate Modified by Chensong Zhang on 05/09/2015: Clean up for stopping types

Definition at line 724 of file KryPvfgmres.c.

9.57.2.2 fasp_solver_dbsr_pvfgmres()

```
INT fasp_solver_dbsr_pvfgmres (
            dBSRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Solve "Ax=b" using PFGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during iteration and flexible preconditioner can be used.

Parameters

A	Pointer to dCSRmat: coefficient matrix
b	Pointer to dvector: right hand side
x	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type – DOES not support this parameter
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

02/05/2012

Modified by Chunsheng Feng on 07/22/2013: Add adaptive memory allocate Modified by Chensong Zhang on 05/09/2015: Clean up for stopping types

Definition at line 394 of file KryPvfgmres.c.

9.57.2.3 fasp_solver_dcsr_pvfgmres()

```
INT fasp_solver_dcsr_pvfgmres (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Solve "Ax=b" using PFGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during iteration and flexible preconditioner can be used.

Parameters

A	Pointer to dCSRmat: coefficient matrix
b	Pointer to dvector: right hand side
x	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type – DOES not support this parameter
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

01/04/2012

Modified by Chunsheng Feng on 07/22/2013: Add adaptive memory allocate Modified by Chensong Zhang on 05/09/2015: Clean up for stopping types

Definition at line 67 of file KryPvfgmres.c.

9.57.2.4 fasp_solver_pvfgmres()

```
INT fasp_solver_pvfgmres (
            mxv_matfree * mf,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Solve "Ax=b" using PFGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during iteration and flexible preconditioner can be used.

Parameters

mf	Pointer to mxv_matfree: spmv operation
b	Pointer to dvector: right hand side
x	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type – DOES not support this parameter
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

01/04/2012

Modified by Feiteng Huang on 09/26/2012: matrix free Modified by Chunsheng Feng on 07/22/2013: Add adapt memory allocate

Definition at line 1051 of file KryPvfgmres.c.

9.58 KryPvgmres.c File Reference

Krylov subspace methods – Preconditioned variable-restart GMRes.

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

Functions

- INT fasp_solver_dcsr_pvgmres (dCSRmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Right preconditioned GMRES method in which the restart parameter can be adaptively modified during iteration.
- INT fasp_solver_dblc_pvgmres (dBLCmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Right preconditioned GMRES method in which the restart parameter can be adaptively modified during iteration.
- INT fasp_solver_dbsr_pvgmres (dBSRmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Right preconditioned GMRES method in which the restart parameter can be adaptively modified during iteration.
- INT fasp_solver_dstr_pvgmres (dSTRmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Right preconditioned GMRES method in which the restart parameter can be adaptively modified during iteration.
- INT fasp_solver_pvgmres (mxv_matfree ∗mf, dvector ∗b, dvector ∗x, precond ∗pc, const REAL tol, const INT MaxIt, SHORT restart, const SHORT stop_type, const SHORT prtlvl)

  Solve "Ax=b" using PGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during iteration.

9.58.1 Detailed Description

Krylov subspace methods – Preconditioned variable-restart GMRes.

Note

This file contains Level-3 (Kry) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, BlaArray.c, Bla↩
SpmvBLC.c, BlaSpmvBSR.c, BlaSpmvCSR.c, and BlaSpmvSTR.c
See KrySPvgmres.c for a safer version

Reference: 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.

TODO: Use one single function for all! –Chensong

9.58.2 Function Documentation

9.58.2.1 fasp_solver_dblc_pvgmres()

```
INT fasp_solver_dblc_pvgmres (
            dBLCmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Right preconditioned GMRES method in which the restart parameter can be adaptively modified during iteration.

Parameters

A	Pointer to dCSRmat: coefficient matrix
b	Pointer to dvector: right hand side
x	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/05/2013

Definition at line 408 of file KryPvgmres.c.

9.58.2.2 fasp_solver_dbsr_pvgmres()

```
INT fasp_solver_dbsr_pvgmres (
            dBSRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Right preconditioned GMRES method in which the restart parameter can be adaptively modified during iteration.

Parameters

A	Pointer to dCSRmat: coefficient matrix
b	Pointer to dvector: right hand side
x	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou

Date

12/21/2011

Modified by Chensong Zhang on 04/06/2013: Add stop type support

Definition at line 753 of file KryPvgmres.c.

9.58.2.3 fasp_solver_dcsr_pvgmres()

```
INT fasp_solver_dcsr_pvgmres (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Right preconditioned GMRES method in which the restart parameter can be adaptively modified during iteration.

Parameters

A	Pointer to dCSRmat: coefficient matrix
b	Pointer to dvector: right hand side
x	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou

Date

2010/12/14

Modified by Chensong Zhang on 04/06/2013: Add stop type support Modified by Chunsheng Feng on 07/22/2013: Add adapt memory allocate

Definition at line 66 of file KryPvgmres.c.

9.58.2.4 fasp_solver_dstr_pvgmres()

```
INT fasp_solver_dstr_pvgmres (
            dSTRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Right preconditioned GMRES method in which the restart parameter can be adaptively modified during iteration.

Parameters

A	Pointer to dCSRmat: coefficient matrix
b	Pointer to dvector: right hand side
x	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou

Date

2010/12/14

Modified by Chensong Zhang on 04/06/2013: Add stop type support

Definition at line 1098 of file KryPvgmres.c.

9.58.2.5 fasp_solver_pvgmres()

```
INT fasp_solver_pvgmres (
            mxv_matfree * mf,
            dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            SHORT restart,
            const SHORT stop_type,
            const SHORT prtlvl )
```

Solve "Ax=b" using PGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during iteration.

Parameters

mf	Pointer to mxv_matfree: spmv operation
b	Pointer to dvector: right hand side
x	Pointer to dvector: unknowns
pc	Pointer to precond: structure of precondition
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
stop_type	Stopping criteria type – DOES not support this parameter
prtlvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou

Date

2010/12/14

Modified by Feiteng Huang on 09/26/2012: matrix free Modified by Chunsheng Feng on 07/22/2013: Add adapt memory allocate

Definition at line 1443 of file KryPvgmres.c.

9.59 KrySPbcgs.c File Reference

Krylov subspace methods – Preconditioned BiCGstab with safety net.

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

Functions

- INT fasp_solver_dcsr_spbcgs (const dCSRmat *A, const dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT StopType, const SHORT PrtLvl)

  Preconditioned BiCGstab method for solving Au=b with safety net.
- INT fasp_solver_dbsr_spbcgs (const dBSRmat *A, const dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT StopType, const SHORT PrtLvl)

  Preconditioned BiCGstab method for solving Au=b with safety net.
- INT fasp_solver_dblc_spbcgs (const dBLCmat *A, const dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT StopType, const SHORT PrtLvl)

  Preconditioned BiCGstab method for solving Au=b with safety net.
- INT fasp_solver_dstr_spbcgs (const dSTRmat *A, const dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT StopType, const SHORT PrtLvl)

  Preconditioned BiCGstab method for solving Au=b with safety net.

9.59.1 Detailed Description

Krylov subspace methods – Preconditioned BiCGstab with safety net.

Note

This file contains Level-3 (Kry) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, AuxVector.c, BlaArray.c, BlaSpmvBLC.c, BlaSpmvBSR.c, BlaSpmvCSR.c, and BlaSpmvSTR.c
See KryPbcgs.c for a version without safety net

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

TODO: Use one single function for all! –Chensong

9.59.2 Function Documentation

9.59.2.1 fasp_solver_dblc_spbcgs()

```
INT fasp_solver_dblc_spbcgs (
            const dBLCmat * A,
            const dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT StopType,
            const SHORT PrtLvl )
```

Preconditioned BiCGstab method for solving Au=b with safety net.

Parameters

A	Pointer to dBLCmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to the structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
StopType	Stopping criteria type
PrtLvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

03/31/2013

Definition at line 835 of file KrySPbcgs.c.

9.59.2.2 fasp_solver_dbsr_spbcgs()

```
INT fasp_solver_dbsr_spbcgs (
            const dBSRmat * A,
            const dvector * b,
            dvector * u,
```

```
precond * pc,
const REAL tol,
const INT MaxIt,
const SHORT StopType,
const SHORT PrtLvl )
```

Preconditioned BiCGstab method for solving Au=b with safety net.

Parameters

A	Pointer to dBSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to the structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
StopType	Stopping criteria type
PrtLvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

03/31/2013

Definition at line 447 of file KrySPbcgs.c.

9.59.2.3 fasp_solver_dcsr_spbcgs()

```
INT fasp_solver_dcsr_spbcgs (
            const dCSRmat * A,
            const dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT StopType,
            const SHORT PrtLvl )
```

Preconditioned BiCGstab method for solving Au=b with safety net.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to the structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
StopType	Stopping criteria type
PrtLvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

03/31/2013

Definition at line 59 of file KrySPbcgs.c.

9.59.2.4 fasp_solver_dstr_spbcgs()

```
INT fasp_solver_dstr_spbcgs (
            const dSTRmat * A,
            const dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT StopType,
            const SHORT PrtLvl )
```

Preconditioned BiCGstab method for solving Au=b with safety net.

Parameters

A	Pointer to dSTRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to the structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
StopType	Stopping criteria type
PrtLvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

03/31/2013

Definition at line 1223 of file KrySPbcgs.c.

9.60 KrySPcg.c File Reference

Krylov subspace methods – Preconditioned CG with safety net.

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

Functions

- INT fasp_solver_dcsr_spcg (const dCSRmat ∗A, const dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT StopType, const SHORT PrtLvl)

  Preconditioned conjugate gradient method for solving Au=b with safety net.
- INT fasp_solver_dblc_spcg (const dBLCmat ∗A, const dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT StopType, const SHORT PrtLvl)

  Preconditioned conjugate gradient method for solving Au=b with safety net.
- INT fasp_solver_dstr_spcg (const dSTRmat ∗A, const dvector ∗b, dvector ∗u, precond ∗pc, const REAL tol, const INT MaxIt, const SHORT StopType, const SHORT PrtLvl)

  Preconditioned conjugate gradient method for solving Au=b with safety net.

9.60.1 Detailed Description

Krylov subspace methods – Preconditioned CG with safety net.

Note

This file contains Level-3 (Kry) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, AuxVector.c, BlaArray.c, BlaSpmvBLC.c, BlaSpmvCSR.c, BlaSpmvSTR.c, and BlaVector.c
See KryPcg.c for a version without safety net

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

TODO: Use one single function for all! –Chensong

9.60.2 Function Documentation

9.60.2.1 fasp_solver_dblc_spcg()

```
INT fasp_solver_dblc_spcg (
            const dBLCmat * A,
            const dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT StopType,
            const SHORT PrtLvl )
```

Preconditioned conjugate gradient method for solving Au=b with safety net.

Parameters

A	Pointer to dBLCmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to the structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
StopType	Stopping criteria type
PrtLvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

03/28/2013

Definition at line 389 of file KrySPcg.c.

9.60.2.2 fasp_solver_dcsr_spcg()

```
INT fasp_solver_dcsr_spcg (
            const dCSRmat * A,
            const dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT StopType,
            const SHORT PrtLvl )
```

Preconditioned conjugate gradient method for solving Au=b with safety net.

Parameters

A	Pointer to dCSRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
pc	Pointer to the structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
StopType	Stopping criteria type
PrtLvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

03/28/2013

Definition at line 59 of file KrySPcg.c.

9.60.2.3 fasp_solver_dstr_spcg()

```
INT fasp_solver_dstr_spcg (
            const dSTRmat * A,
            const dvector * b,
            dvector * u,
```

```
precond * pc,
const REAL tol,
const INT MaxIt,
const SHORT StopType,
const SHORT PrtLvl )
```

Preconditioned conjugate gradient method for solving Au=b with safety net.

Parameters

A	Pointer to dSTRmat: the coefficient matrix
b	Pointer to dvector: the right hand side
u	Pointer to dvector: the unknowns
MaxIt	Maximal number of iterations
tol	Tolerance for stopping
pc	Pointer to the structure of precondition (precond)
StopType	Stopping criteria type
PrtLvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

03/28/2013

Definition at line 719 of file KrySPcg.c.

9.61 KrySPgmres.c File Reference

Krylov subspace methods – Preconditioned GMRes with safety net.

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

Functions

- INT fasp_solver_dcsr_spgmres (const dCSRmat *A, const dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, SHORT restart, const SHORT StopType, const SHORT PrtLvl)

  Preconditioned GMRES method for solving Au=b with safe-guard.
- INT fasp_solver_dblc_spgmres (const dBLCmat *A, const dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, SHORT restart, const SHORT StopType, const SHORT PrtLvl)

  Preconditioned GMRES method for solving Au=b with safe-guard.
- INT fasp_solver_dbsr_spgmres (const dBSRmat *A, const dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, SHORT restart, const SHORT StopType, const SHORT PrtLvl)

  Preconditioned GMRES method for solving Au=b with safe-guard.
- INT fasp_solver_dstr_spgmres (const dSTRmat *A, const dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, SHORT restart, const SHORT StopType, const SHORT PrtLvl)

  Preconditioned GMRES method for solving Au=b with safe-guard.

9.61.1 Detailed Description

Krylov subspace methods – Preconditioned GMRes with safety net.

Note

This file contains Level-3 (Kry) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, AuxVector.c, BlaArray.c, BlaSpmvBLC.c, BlaSpmvBSR.c, BlaSpmvCSR.c, and BlaSpmvSTR.c
See also pgmres.c for a variable restarting version.
See KryPgmres.c for a version without safety net

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

TODO: Use one single function for all! –Chensong

9.61.2 Function Documentation

9.61.2.1 fasp_solver_dblc_spgmres()

```
INT fasp_solver_dblc_spgmres (
            const dBLCmat * A,
            const dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            SHORT restart,
            const SHORT StopType,
            const SHORT PrtLvl )
```

Preconditioned GMRES method for solving Au=b with safe-guard.

Parameters

A	Pointer to dBLCmat: coefficient matrix
b	Pointer to dvector: right hand side
x	Pointer to dvector: unknowns
pc	Pointer to structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
StopType	Stopping criteria type
PrtLvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/05/2013

Definition at line 406 of file KrySPgmres.c.

9.61.2.2 fasp_solver_dbsr_spgmres()

```
INT fasp_solver_dbsr_spgmres (
            const dBSRmat * A,
            const dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            SHORT restart,
            const SHORT StopType,
            const SHORT PrtLvl )
```

Preconditioned GMRES method for solving Au=b with safe-guard.

Parameters

A	Pointer to dBSRmat: coefficient matrix
b	Pointer to dvector: right hand side
x	Pointer to dvector: unknowns
pc	Pointer to structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
StopType	Stopping criteria type
PrtLvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/05/2013

Definition at line 747 of file KrySPgmres.c.

9.61.2.3 fasp_solver_dcsr_spgmres()

```
INT fasp_solver_dcsr_spgmres (
            const dCSRmat * A,
            const dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            SHORT restart,
            const SHORT StopType,
            const SHORT PrtLvl )
```

Preconditioned GMRES method for solving Au=b with safe-guard.

Parameters

A	Pointer to dCSRmat: coefficient matrix
b	Pointer to dvector: right hand side
x	Pointer to dvector: unknowns
pc	Pointer to structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
StopType	Stopping criteria type
PrtLvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/05/2013

Modified by Chunsheng Feng on 07/22/2013: Add adapt memory allocate

Definition at line 65 of file KrySPgmres.c.

9.61.2.4 fasp_solver_dstr_spgmres()

```
INT fasp_solver_dstr_spgmres (
            const dSTRmat * A,
            const dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            SHORT restart,
            const SHORT StopType,
            const SHORT PrtLvl )
```

Preconditioned GMRES method for solving Au=b with safe-guard.

Parameters

A	Pointer to dSTRmat: coefficient matrix
b	Pointer to dvector: right hand side
x	Pointer to dvector: unknowns
pc	Pointer to structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
StopType	Stopping criteria type
PrtLvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/05/2013

Definition at line 1088 of file KrySPgmres.c.

9.62 KrySPminres.c File Reference

Krylov subspace methods – Preconditioned MINRES with safety net.

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

Functions

- INT fasp_solver_dcsr_spminres (const dCSRmat *A, const dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT StopType, const SHORT PrtLvl)

  A preconditioned minimal residual (Minres) method for solving Au=b with safety net.
- INT fasp_solver_dblc_spminres (const dBLCmat *A, const dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT StopType, const SHORT PrtLvl)

  A preconditioned minimal residual (Minres) method for solving Au=b with safety net.
- INT fasp_solver_dstr_spminres (const dSTRmat *A, const dvector *b, dvector *u, precond *pc, const REAL tol, const INT MaxIt, const SHORT StopType, const SHORT PrtLvl)

  A preconditioned minimal residual (Minres) method for solving Au=b with safety net.

9.62.1 Detailed Description

Krylov subspace methods – Preconditioned MINRES with safety net.

Note

This file contains Level-3 (Kry) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, AuxVector.c, BlaArray.c, BlaSpmvBLC.c, BlaSpmvCSR.c, and BlaSpmvSTR.c
See KryPminres.c for a version without safety net

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

TODO: Use one single function for all! –Chensong

9.62.2 Function Documentation

9.62.2.1 fasp_solver_dblc_spminres()

```
INT fasp_solver_dblc_spminres (
            const dBLCmat * A,
            const dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT StopType,
            const SHORT PrtLvl )
```

A preconditioned minimal residual (Minres) method for solving Au=b with safety net.

Parameters

A	Pointer to dBLCmat: coefficient matrix
b	Pointer to dvector: right hand side
u	Pointer to dvector: unknowns
pc	Pointer to structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
StopType	Stopping criteria type
PrtLvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/09/2013

Definition at line 507 of file KrySPminres.c.

9.62.2.2 fasp_solver_dcsr_spminres()

```
INT fasp_solver_dcsr_spminres (
            const dCSRmat * A,
            const dvector * b,
            dvector * u,
```

```
precond * pc,
const REAL tol,
const INT MaxIt,
const SHORT StopType,
const SHORT PrtLvl )
```

A preconditioned minimal residual (Minres) method for solving Au=b with safety net.

Parameters

A	Pointer to dCSRmat: coefficient matrix
b	Pointer to dvector: right hand side
u	Pointer to dvector: unknowns
pc	Pointer to structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
StopType	Stopping criteria type
PrtLvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/09/2013

Definition at line 59 of file KrySPminres.c.

9.62.2.3 fasp_solver_dstr_spminres()

```
INT fasp_solver_dstr_spminres (
            const dSTRmat * A,
            const dvector * b,
            dvector * u,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            const SHORT StopType,
            const SHORT PrtLvl )
```

A preconditioned minimal residual (Minres) method for solving Au=b with safety net.

Parameters

A	Pointer to dSTRmat: coefficient matrix
b	Pointer to dvector: right hand side
u	Pointer to dvector: unknowns
MaxIt	Maximal number of iterations
tol	Tolerance for stopping
pc	Pointer to structure of precondition (precond)
StopType	Stopping criteria type
PrtLvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/09/2013

Definition at line 955 of file KrySPminres.c.

9.63 KrySPvgmres.c File Reference

Krylov subspace methods – Preconditioned variable-restart GMRes with safety net.

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

Functions

- INT fasp_solver_dcsr_spvgmres (const dCSRmat *A, const dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, SHORT restart, const SHORT StopType, const SHORT PrtLvl)

    Solve "Ax=b" using PGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during iteration.

- INT fasp_solver_dblc_spvgmres (const dBLCmat *A, const dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, SHORT restart, const SHORT StopType, const SHORT PrtLvl)

    Preconditioned GMRES method for solving Au=b.

- INT fasp_solver_dbsr_spvgmres (const dBSRmat *A, const dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, SHORT restart, const SHORT StopType, const SHORT PrtLvl)

    Solve "Ax=b" using PGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during iteration.

- INT fasp_solver_dstr_spvgmres (const dSTRmat *A, const dvector *b, dvector *x, precond *pc, const REAL tol, const INT MaxIt, SHORT restart, const SHORT StopType, const SHORT PrtLvl)

    Solve "Ax=b" using PGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during iteration.

9.63.1 Detailed Description

Krylov subspace methods – Preconditioned variable-restart GMRes with safety net.

Note

This file contains Level-3 (Kry) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, AuxVector.c, BlaArray.c, BlaSpmvBLC.c, BlaSpmvBSR.c, BlaSpmvCSR.c, and BlaSpmvSTR.c
See KryPvgmres.c a version without safety net

Reference: 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.

TODO: Use one single function for all! –Chensong

9.63.2 Function Documentation

9.63.2.1 fasp_solver_dblc_spvgmres()

```
INT fasp_solver_dblc_spvgmres (
              const dBLCmat * A,
              const dvector * b,
              dvector * x,
              precond * pc,
              const REAL tol,
              const INT MaxIt,
              SHORT restart,
              const SHORT StopType,
              const SHORT PrtLvl )
```

Preconditioned GMRES method for solving Au=b.

Parameters

A	Pointer to dBLCmat: coefficient matrix
b	Pointer to dvector: right hand side
x	Pointer to dvector: unknowns
pc	Pointer to structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
StopType	Stopping criteria type
PrtLvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/06/2013

Definition at line 445 of file KrySPvgmres.c.

9.63.2.2 fasp_solver_dbsr_spvgmres()

```
INT fasp_solver_dbsr_spvgmres (
            const dBSRmat * A,
            const dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            SHORT restart,
            const SHORT StopType,
            const SHORT PrtLvl )
```

Solve "Ax=b" using PGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during iteration.

Parameters

A	Pointer to dBSRmat: coefficient matrix
b	Pointer to dvector: right hand side
x	Pointer to dvector: unknowns
pc	Pointer to structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
StopType	Stopping criteria type
PrtLvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/06/2013

Definition at line 824 of file KrySPvgmres.c.

9.63.2.3 fasp_solver_dcsr_spvgmres()

```
INT fasp_solver_dcsr_spvgmres (
            const dCSRmat * A,
            const dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            SHORT restart,
            const SHORT StopType,
            const SHORT PrtLvl )
```

Solve "Ax=b" using PGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during iteration.

Parameters

A	Pointer to dCSRmat: coefficient matrix
b	Pointer to dvector: right hand side
x	Pointer to dvector: unknowns
pc	Pointer to structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
StopType	Stopping criteria type
PrtLvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/06/2013

Modified by Chunsheng Feng on 07/22/2013: Add adapt memory allocate

Definition at line 67 of file KrySPvgmres.c.

9.63.2.4 fasp_solver_dstr_spvgmres()

```
INT fasp_solver_dstr_spvgmres (
            const dSTRmat * A,
            const dvector * b,
            dvector * x,
            precond * pc,
            const REAL tol,
            const INT MaxIt,
            SHORT restart,
            const SHORT StopType,
            const SHORT PrtLvl )
```

Solve "Ax=b" using PGMRES(right preconditioned) iterative method in which the restart parameter can be adaptively modified during iteration.

Parameters

A	Pointer to dSTRmat: coefficient matrix
b	Pointer to dvector: right hand side
x	Pointer to dvector: unknowns
pc	Pointer to structure of precondition (precond)
tol	Tolerance for stopping
MaxIt	Maximal number of iterations
restart	Restarting steps
StopType	Stopping criteria type
PrtLvl	How much information to print out

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

04/06/2013

Definition at line 1203 of file KrySPvgmres.c.

9.64 PreAMGCoarsenCR.c File Reference

Coarsening with Brannick-Falgout strategy.

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

Functions

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

  CR coarsening.

9.64.1 Detailed Description

Coarsening with Brannick-Falgout strategy.

Note

This file contains Level-4 (Pre) functions. It requires: AuxMemory.c, AuxThreads.c, and ItrSmootherCSRcr.c

// TODO: Not completed! –Chensong // TODO: Fix Doxygen issues in this file! –Chensong

9.64.2 Function Documentation

9.64.2.1 fasp_amg_coarsening_cr()

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

CR coarsening.

Parameters

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

Returns

Number of coarse level points

Author

James Brannick

Date

04/21/2010

Note

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

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

Definition at line 61 of file PreAMGCoarsenCR.c.

9.65 PreAMGCoarsenRS.c File Reference

Coarsening with a modified Ruge-Stuben strategy.

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

Functions

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

  Standard and aggressive coarsening schemes.

9.65.1 Detailed Description

Coarsening with a modified Ruge-Stuben strategy.

Note

This file contains Level-4 (Pre) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, AuxThreads.c, AuxVector.c, BlaSparseCSR.c, and PreAMGCoarsenCR.c

Reference: 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.

Warning

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

9.65.2 Function Documentation

9.65.2.1 fasp_amg_coarsening_rs()

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

Standard and aggressive coarsening schemes.

Parameters

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

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Xuehai Huang, Chensong Zhang, Xiaozhe Hu, Ludmil Zikatanov

Date

09/06/2010

Note

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

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

Definition at line 75 of file PreAMGCoarsenRS.c.

9.66 PreAMGInterp.c File Reference

Direct and standard interpolations for classical 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_interp_trunc (dCSRmat *P, AMG_param *param)

  Truncation step for prolongation operators.

9.66.1 Detailed Description

Direct and standard interpolations for classical AMG.

Note

This file contains Level-4 (Pre) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, AuxThreads.c, and PreAMGInterpEM.c

Reference: U. Trottenberg, C. W. Oosterlee, and A. Schuller Multigrid (Appendix A: An Intro to Algebraic Multigrid) Academic Press Inc., San Diego, CA, 2001 With contributions by A. Brandt, P. Oswald and K. Stuben.

9.66.2 Function Documentation

9.66.2.1 fasp_amg_interp()

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

Generate interpolation operator P.

Parameters

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

Author

Xuehai Huang, Chensong Zhang

Date

04/04/2010

Modified by Xiaozhe Hu on 05/23/2012: add S as input Modified by Chensong Zhang on 09/12/2012: clean up and debug interp_RS Modified by Chensong Zhang on 05/14/2013: reconstruct the code

Definition at line 62 of file PreAMGInterp.c.

9.66.2.2 fasp_amg_interp_trunc()

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

Truncation step for prolongation operators.

Parameters

P	Prolongation (input: full, output: truncated)
param	Pointer to AMG_param: AMG parameters

Author

Chensong Zhang

Date

05/14/2013

Originally by Xuehai Huang, Chensong Zhang on 01/31/2009 Modified by Chunsheng Feng, Xiaoqiang Yue on 05/23/2012: add OMP support Modified by Chensong Zhang on 05/14/2013: rewritten

Definition at line 117 of file PreAMGInterp.c.

9.67 PreAMGInterpEM.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.

9.67.1 Detailed Description

Interpolation operators for AMG based on energy-min.

Note

This file contains Level-4 (Pre) functions. It requires: AuxArray.c, AuxMemory.c, AuxThreads.c, AuxVector.c, Bla↩ SmallMatLU.c, BlaSparseCSR.c, KryPcg.c, and PreCSR.c

Reference: J. Xu and L. Zikatanov On An Energy Minimizing Basis in Algebraic Multigrid Methods, Computing and visualization in sciences, 2003

9.67.2 Function Documentation

9.67.2.1 fasp_amg_interp_em()

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

Energy-min interpolation.

Parameters

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

Author

Shuo Zhang, Xuehai Huang

Date

04/04/2010

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

Definition at line 63 of file PreAMGInterpEM.c.

9.68 PreAMGSetupCR.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.

9.68.1 Detailed Description

Brannick-Falgout compatible relaxation based AMG: SETUP phase.

Note

This file contains Level-4 (Pre) functions. It requires: AuxMessage.c, AuxTiming.c, AuxVector.c, and PreAMG↩ CoarsenCR.c
Setup A, P, R and levels using the Compatible Relaxation coarsening for classic AMG interpolation

Reference: J. Brannick and R. Falgout Compatible relaxation and coarsening in AMG

TODO: Not working. Need to be fixed. –Chensong

9.68.2 Function Documentation

9.68.2.1 fasp_amg_setup_cr()

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

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

Parameters

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

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

James Brannick

Date

04/21/2010

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

Definition at line 48 of file PreAMGSetupCR.c.

9.69 PreAMGSetupRS.c File Reference

Ruge-Stuben AMG: SETUP phase.

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

Functions

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

  Setup phase of Ruge and Stuben's classic AMG.

9.69.1 Detailed Description

Ruge-Stuben AMG: SETUP phase.

Note

This file contains Level-4 (Pre) functions. It requires: AuxMemory.c, AuxMessage.c, AuxTiming.c, AuxVector.c, BlaILUSetupCSR.c, BlaSchwarzSetup.c, BlaSparseCSR.c, BlaSpmvCSR.c, PreAMGCoarsenRS.c, PreAMGInterp.c, and PreMGRecurAMLI.c

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

9.69.2 Function Documentation

9.69.2.1 fasp_amg_setup_rs()

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

Setup phase of Ruge and Stuben's classic AMG.

Parameters

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

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Chensong Zhang

Date

05/09/2010

Modified by Xiaozhe Hu on 01/23/2011: add AMLI cycle. Modified by Xiaozhe Hu on 04/24/2013: aggressive coarsening. Modified by Chensong Zhang on 09/23/2014: check coarse spaces.

Definition at line 51 of file PreAMGSetupRS.c.

9.70 PreAMGSetupSA.c File Reference

Smoothed aggregation AMG: SETUP phase.

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

Functions

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

    Set up phase of smoothed aggregation AMG.

9.70.1 Detailed Description

Smoothed aggregation AMG: SETUP phase.

Note

This file contains Level-4 (Pre) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, AuxThreads.c, AuxTiming.c, AuxVector.c, BlaILUSetupCSR.c, BlaSchwarzSetup.c, BlaSparseCSR.c, BlaSpmvCSR.c, and Pre←MGRecurAMLI.c
Setup A, P, PT and levels using the unsmoothed aggregation algorithm

Reference: P. Vanek, J. Madel and M. Brezina Algebraic Multigrid on Unstructured Meshes, 1994

9.70.2 Function Documentation

9.70.2.1 fasp_amg_setup_sa()

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

Set up phase of smoothed aggregation AMG.

Parameters

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

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Xiaozhe Hu

Date

09/29/2009

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

Definition at line 62 of file PreAMGSetupSA.c.

9.71 PreAMGSetupSABSR.c File Reference

Smoothed aggregation AMG: SETUP phase (for BSR matrices)

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

Functions

- SHORT fasp_amg_setup_sa_bsr (AMG_data_bsr ∗mgl, AMG_param ∗param)

  Set up phase of smoothed aggregation AMG (BSR format)

9.71.1 Detailed Description

Smoothed aggregation AMG: SETUP phase (for BSR matrices)

Note

This file contains Level-4 (Pre) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, AuxTiming.c, AuxVector.c, BlaFormat.c, BlaILUSetupBSR.c, BlaSmallMat.c, BlaSparseBLC.c, BlaSparseBSR.c, BlaSparseC←SR.c, BlaSpmvBSR.c, and BlaSpmvCSR.c
Setup A, P, PT and levels using the unsmoothed aggregation algorithm

Reference: P. Vanek, J. Madel and M. Brezina Algebraic Multigrid on Unstructured Meshes, 1994

9.71.2 Function Documentation

9.71.2.1 fasp_amg_setup_sa_bsr()

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

Set up phase of smoothed aggregation AMG (BSR format)

Parameters

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

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Xiaozhe Hu

Date

05/26/2014

Definition at line 58 of file PreAMGSetupSABSR.c.

9.72 PreAMGSetupUA.c File Reference

Unsmoothed aggregation AMG: SETUP phase.

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

Functions

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

  Set up phase of unsmoothed aggregation AMG.

9.72.1 Detailed Description

Unsmoothed aggregation AMG: SETUP phase.

Note

This file contains Level-4 (Pre) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, AuxTiming.c, AuxVector.c, BlaILUSetupCSR.c, BlaSchwarzSetup.c, BlaSparseCSR.c, BlaSpmvCSR.c, and PreMGRecurAM↩LI.c
Setup A, P, PT and levels using the unsmoothed aggregation algorithm

Reference: P. Vanek, J. Madel and M. Brezina Algebraic Multigrid on Unstructured Meshes, 1994

9.72.2 Function Documentation

9.72.2.1 fasp_amg_setup_ua()

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

Set up phase of unsmoothed aggregation AMG.

Parameters

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

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Xiaozhe Hu

Date

12/28/2011

Definition at line 54 of file PreAMGSetupUA.c.

9.73 PreAMGSetupUABSR.c File Reference

Unsmoothed aggregation AMG: SETUP phase (for BSR matrices)

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

Functions

- SHORT fasp_amg_setup_ua_bsr (AMG_data_bsr *mgl, AMG_param *param)
  Set up phase of unsmoothed aggregation AMG (BSR format)

9.73.1 Detailed Description

Unsmoothed aggregation AMG: SETUP phase (for BSR matrices)

Note

This file contains Level-4 (Pre) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, AuxTiming.c, AuxVector.c, BlaFormat.c, BlaILUSetupBSR.c, BlaSparseBLC.c, BlaSparseBSR.c, BlaSparseCSR.c, BlaSpmvBSR.c, BlaSpmvCSR.c, and PreDataInit.c
Setup A, P, PT and levels using the unsmoothed aggregation algorithm

Reference: P. Vanek, J. Madel and M. Brezina Algebraic Multigrid on Unstructured Meshes, 1994

9.73.2 Function Documentation

9.73.2.1 fasp_amg_setup_ua_bsr()

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

Set up phase of unsmoothed aggregation AMG (BSR format)

Parameters

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

Returns

FASP_SUCCESS if successed; otherwise, error information.

Author

Xiaozhe Hu

Date

03/16/2012

Definition at line 54 of file PreAMGSetupUABSR.c.

9.74 PreBLC.c File Reference

Preconditioners for dBLCmat matrices.

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

Functions

- void fasp_precond_block_diag_3 (REAL *r, REAL *z, void *data)

  block diagonal preconditioning (3x3 block matrix, each diagonal block is solved exactly)
- void fasp_precond_block_diag_3_amg (REAL *r, REAL *z, void *data)

  block diagonal preconditioning (3x3 block matrix, each diagonal block is solved by AMG)
- void fasp_precond_block_diag_4 (REAL *r, REAL *z, void *data)

  block diagonal preconditioning (4x4 block matrix, each diagonal block is solved exactly)
- void fasp_precond_block_lower_3 (REAL *r, REAL *z, void *data)

  block lower triangular preconditioning (3x3 block matrix, each diagonal block is solved exactly)
- void fasp_precond_block_lower_3_amg (REAL *r, REAL *z, void *data)

  block lower triangular preconditioning (3x3 block matrix, each diagonal block is solved by AMG)
- void fasp_precond_block_lower_4 (REAL *r, REAL *z, void *data)

  block lower triangular preconditioning (4x4 block matrix, each diagonal block is solved exactly)
- void fasp_precond_block_upper_3 (REAL *r, REAL *z, void *data)

  block upper triangular preconditioning (3x3 block matrix, each diagonal block is solved exactly)
- void fasp_precond_block_upper_3_amg (REAL *r, REAL *z, void *data)

  block upper triangular preconditioning (3x3 block matrix, each diagonal block is solved AMG)
- void fasp_precond_block_SGS_3 (REAL *r, REAL *z, void *data)

  block symmetric GS preconditioning (3x3 block matrix, each diagonal block is solved exactly)
- void fasp_precond_block_SGS_3_amg (REAL *r, REAL *z, void *data)

  block symmetric GS preconditioning (3x3 block matrix, each diagonal block is solved exactly)
- void fasp_precond_sweeping (REAL *r, REAL *z, void *data)

  sweeping preconditioner for Maxwell equations

9.74.1 Detailed Description

Preconditioners for dBLCmat matrices.

Note

This file contains Level-4 (Pre) functions. It requires: AuxArray.c, AuxMemory.c, AuxVector.c, BlaSpmvCSR.c, and PreMGCycle.c

TODO: Need to be cleaned up. –Chensong

9.74.2 Function Documentation

9.74.2.1 fasp_precond_block_diag_3()

```
void fasp_precond_block_diag_3 (
            REAL * r,
            REAL * z,
            void * data )
```

block diagonal preconditioning (3x3 block matrix, each diagonal block is solved exactly)

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

07/10/2014

Definition at line 36 of file PreBLC.c.

9.74.2.2 fasp_precond_block_diag_3_amg()

```
void fasp_precond_block_diag_3_amg (
            REAL * r,
            REAL * z,
            void * data )
```

block diagonal preconditioning (3x3 block matrix, each diagonal block is solved by AMG)

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

07/10/2014

Definition at line 120 of file PreBLC.c.

9.74.2.3 fasp_precond_block_diag_4()

```
void fasp_precond_block_diag_4 (
            REAL * r,
            REAL * z,
            void * data )
```

block diagonal preconditioning (4x4 block matrix, each diagonal block is solved exactly)

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

07/10/2014

Definition at line 185 of file PreBLC.c.

9.74.2.4 fasp_precond_block_lower_3()

```
void fasp_precond_block_lower_3 (
            REAL * r,
            REAL * z,
            void * data )
```

block lower triangular preconditioning (3x3 block matrix, each diagonal block is solved exactly)

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

07/10/2014

Definition at line 281 of file PreBLC.c.

9.74.2.5 fasp_precond_block_lower_3_amg()

```
void fasp_precond_block_lower_3_amg (
            REAL * r,
            REAL * z,
            void * data )
```

block lower triangular preconditioning (3x3 block matrix, each diagonal block is solved by AMG)

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

07/10/2014

Definition at line 367 of file PreBLC.c.

9.74.2.6 fasp_precond_block_lower_4()

```
void fasp_precond_block_lower_4 (
            REAL * r,
            REAL * z,
            void * data )
```

block lower triangular preconditioning (4x4 block matrix, each diagonal block is solved exactly)

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

07/10/2014

Definition at line 441 of file PreBLC.c.

9.74.2.7 fasp_precond_block_SGS_3()

```
void fasp_precond_block_SGS_3 (
            REAL * r,
            REAL * z,
            void * data )
```

block symmetric GS preconditioning (3x3 block matrix, each diagonal block is solved exactly)

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

02/19/2015

Definition at line 709 of file PreBLC.c.

9.74.2.8 fasp_precond_block_SGS_3_amg()

```
void fasp_precond_block_SGS_3_amg (
            REAL * r,
            REAL * z,
            void * data )
```

block symmetric GS preconditioning (3x3 block matrix, each diagonal block is solved exactly)

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

02/19/2015

Definition at line 818 of file PreBLC.c.

9.74.2.9 fasp_precond_block_upper_3()

```
void fasp_precond_block_upper_3 (
            REAL * r,
            REAL * z,
            void * data )
```

block upper triangular preconditioning (3x3 block matrix, each diagonal block is solved exactly)

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

02/18/2015

Definition at line 543 of file PreBLC.c.

9.74.2.10 fasp_precond_block_upper_3_amg()

```
void fasp_precond_block_upper_3_amg (
            REAL * r,
            REAL * z,
            void * data )
```

block upper triangular preconditioning (3x3 block matrix, each diagonal block is solved AMG)

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

02/19/2015

Definition at line 629 of file PreBLC.c.

9.74.2.11 fasp_precond_sweeping()

```
void fasp_precond_sweeping (
            REAL * r,
            REAL * z,
            void * data )
```

sweeping preconditioner for Maxwell equations

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

05/01/2014

Definition at line 927 of file PreBLC.c.

9.75 PreBSR.c File Reference

Preconditioners for dBSRmat matrices.

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

Functions

- void fasp_precond_dbsr_diag (REAL ∗r, REAL ∗z, void ∗data)
  Diagonal preconditioner z=inv(D)∗r.
- void fasp_precond_dbsr_diag_nc2 (REAL ∗r, REAL ∗z, void ∗data)
  Diagonal preconditioner z=inv(D)∗r.
- void fasp_precond_dbsr_diag_nc3 (REAL ∗r, REAL ∗z, void ∗data)
  Diagonal preconditioner z=inv(D)∗r.
- void fasp_precond_dbsr_diag_nc5 (REAL ∗r, REAL ∗z, void ∗data)
  Diagonal preconditioner z=inv(D)∗r.
- void fasp_precond_dbsr_diag_nc7 (REAL ∗r, REAL ∗z, void ∗data)
  Diagonal preconditioner z=inv(D)∗r.
- void fasp_precond_dbsr_ilu (REAL ∗r, REAL ∗z, void ∗data)
  ILU preconditioner.
- void fasp_precond_dbsr_ilu_mc_omp (REAL ∗r, REAL ∗z, void ∗data)
  Multi-thread Parallel ILU preconditioner based on graph coloring.
- void fasp_precond_dbsr_ilu_ls_omp (REAL ∗r, REAL ∗z, void ∗data)
  Multi-thread Parallel ILU preconditioner based on level schedule strategy.
- void fasp_precond_dbsr_amg (REAL ∗r, REAL ∗z, void ∗data)
  AMG preconditioner.
- void fasp_precond_dbsr_namli (REAL ∗r, REAL ∗z, void ∗data)
  Nonlinear AMLI-cycle AMG preconditioner.
- void fasp_precond_dbsr_amg_nk (REAL ∗r, REAL ∗z, void ∗data)
  AMG with extra near kernel solve preconditioner.

9.75.1 Detailed Description

Preconditioners for dBSRmat matrices.

Note

This file contains Level-4 (Pre) functions. It requires: AuxArray.c, AuxParam.c, AuxThreads.c, AuxVector.c, Bla←SmallMat.c, BlaSpmvBSR.c, BlaSpmvCSR.c, KrySPcg.c, KrySPvgmres.c, PreMGCycle.c, and PreMGRecurA←MLI.c

9.75.2 Function Documentation

9.75.2.1 fasp_precond_dbsr_amg()

```
void fasp_precond_dbsr_amg (
            REAL * r,
            REAL * z,
            void * data )
```

AMG preconditioner.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

08/07/2011

Definition at line 974 of file PreBSR.c.

9.75.2.2 fasp_precond_dbsr_amg_nk()

```
void fasp_precond_dbsr_amg_nk (
            REAL * r,
            REAL * z,
            void * data )
```

AMG with extra near kernel solve preconditioner.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

05/26/2014

Definition at line 1054 of file PreBSR.c.

9.75.2.3 fasp_precond_dbsr_diag()

```
void fasp_precond_dbsr_diag (
            REAL * r,
            REAL * z,
            void * data )
```

Diagonal preconditioner z=inv(D)*r.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Zhou Zhiyang, Xiaozhe Hu

Date

10/26/2010

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

Note

Works for general nb (Xiaozhe)

Definition at line 49 of file PreBSR.c.

9.75.2.4 fasp_precond_dbsr_diag_nc2()

```
void fasp_precond_dbsr_diag_nc2 (
            REAL * r,
            REAL * z,
            void * data )
```

Diagonal preconditioner z=inv(D)*r.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Zhou Zhiyang, Xiaozhe Hu

Date

11/18/2011

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

Note

Works for 2-component (Xiaozhe)

Definition at line 121 of file PreBSR.c.

9.75.2.5 fasp_precond_dbsr_diag_nc3()

```
void fasp_precond_dbsr_diag_nc3 (
            REAL * r,
            REAL * z,
            void * data )
```

Diagonal preconditioner z=inv(D)∗r.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Zhou Zhiyang, Xiaozhe Hu

Date

01/06/2011

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

Note

Works for 3-component (Xiaozhe)

Definition at line 169 of file PreBSR.c.

9.75.2.6 fasp_precond_dbsr_diag_nc5()

```
void fasp_precond_dbsr_diag_nc5 (
            REAL * r,
            REAL * z,
            void * data )
```

Diagonal preconditioner z=inv(D)*r.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Zhou Zhiyang, Xiaozhe Hu

Date

01/06/2011

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

Note

Works for 5-component (Xiaozhe)

Definition at line 217 of file PreBSR.c.

9.75.2.7 fasp_precond_dbsr_diag_nc7()

```
void fasp_precond_dbsr_diag_nc7 (
            REAL * r,
            REAL * z,
            void * data )
```

Diagonal preconditioner z=inv(D)*r.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Zhou Zhiyang, Xiaozhe Hu

Date

01/06/2011

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

Note

Works for 7-component (Xiaozhe)

Definition at line 265 of file PreBSR.c.

9.75.2.8 fasp_precond_dbsr_ilu()

```
void fasp_precond_dbsr_ilu (
            REAL * r,
            REAL * z,
            void * data )
```

ILU preconditioner.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Shiquan Zhang, Xiaozhe Hu

Date

11/09/2010

Note

Works for general nb (Xiaozhe)

Definition at line 311 of file PreBSR.c.

9.75.2.9 fasp_precond_dbsr_ilu_ls_omp()

```
void fasp_precond_dbsr_ilu_ls_omp (
            REAL * r,
            REAL * z,
            void * data )
```

Multi-thread Parallel ILU preconditioner based on level schedule strategy.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

ZhengLi

Date

12/04/2016

Note

Only works for nb 1, 2, and 3 (Zheng)

Definition at line 767 of file PreBSR.c.

9.75.2.10 fasp_precond_dbsr_ilu_mc_omp()

```
void fasp_precond_dbsr_ilu_mc_omp (
            REAL * r,
            REAL * z,
            void * data )
```

Multi-thread Parallel ILU preconditioner based on graph coloring.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

ZhengLi

Date

12/04/2016

Note

Only works for nb 1, 2, and 3 (Zheng)

Definition at line 569 of file PreBSR.c.

9.75.2.11 fasp_precond_dbsr_namli()

```
void fasp_precond_dbsr_namli (
            REAL * r,
            REAL * z,
            void * data )
```

Nonlinear AMLI-cycle AMG preconditioner.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

02/06/2012

Definition at line 1017 of file PreBSR.c.

9.76 PreCSR.c File Reference

Preconditioners for dCSRmat matrices.

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

Functions

- precond ∗ fasp_precond_setup (const SHORT precond_type, AMG_param ∗amgparam, ILU_param ∗iluparam, dCSRmat ∗A)

    Setup preconditioner interface for iterative methods.
- void fasp_precond_diag (REAL ∗r, REAL ∗z, void ∗data)

    Diagonal preconditioner z=inv(D)∗r.
- void fasp_precond_ilu (REAL ∗r, REAL ∗z, void ∗data)

    ILU preconditioner.
- void fasp_precond_ilu_forward (REAL ∗r, REAL ∗z, void ∗data)

    ILU preconditioner: only forward sweep.
- void fasp_precond_ilu_backward (REAL ∗r, REAL ∗z, void ∗data)

    ILU preconditioner: only backward sweep.
- void fasp_precond_swz (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_namli (REAL ∗r, REAL ∗z, void ∗data)

    Nonlinear AMLI AMG preconditioner.
- void fasp_precond_amg_nk (REAL ∗r, REAL ∗z, void ∗data)

    AMG with extra near kernel solve as preconditioner.
- void fasp_precond_free (const SHORT precond_type, precond ∗pc)

    free preconditioner

9.76.1 Detailed Description

Preconditioners for dCSRmat matrices.

Note

This file contains Level-4 (Pre) functions. It requires: AuxArray.c, AuxMemory.c, AuxParam.c, AuxVector.c, Bla↩ ILUSetupCSR.c, BlaSchwarzSetup.c, BlaSparseCSR.c, BlaSpmvCSR.c, KrySPcg.c, KrySPvgmres.c, PreAMG↩ SetupRS.c, PreAMGSetupSA.c, PreAMGSetupUA.c, PreDataInit.c, PreMGCycle.c, PreMGCycleFull.c, and Pre↩ MGRecurAMLI.c

9.76.2 Function Documentation

9.76.2.1 fasp_precond_amg()

```
void fasp_precond_amg (
            REAL * r,
            REAL * z,
            void * data )
```

AMG preconditioner.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Chensong Zhang

Date

04/06/2010

Definition at line 413 of file PreCSR.c.

9.76.2.2 fasp_precond_amg_nk()

```
void fasp_precond_amg_nk (
            REAL * r,
            REAL * z,
            void * data )
```

AMG with extra near kernel solve as preconditioner.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

05/26/2014

Definition at line 545 of file PreCSR.c.

9.76.2.3 fasp_precond_amli()

```
void fasp_precond_amli (
            REAL * r,
            REAL * z,
            void * data )
```

AMLI AMG preconditioner.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

01/23/2011

Definition at line 479 of file PreCSR.c.

9.76.2.4 fasp_precond_diag()

```
void fasp_precond_diag (
            REAL * r,
            REAL * z,
            void * data )
```

Diagonal preconditioner z=inv(D)∗r.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Chensong Zhang

Date

04/06/2010

Definition at line 172 of file PreCSR.c.

9.76.2.5 fasp_precond_famg()

```
void fasp_precond_famg (
            REAL * r,
            REAL * z,
            void * data )
```

Full AMG preconditioner.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

02/27/2011

Definition at line 446 of file PreCSR.c.

9.76.2.6 fasp_precond_free()

```
void fasp_precond_free (
            const SHORT precond_type,
            precond * pc )
```

free preconditioner

Parameters

precond_type	Preconditioner type
pc	Preconditioner data & fct

Author

Feiteng Huang

Date

12/24/2012

Definition at line 627 of file PreCSR.c.

9.76.2.7 fasp_precond_ilu()

```
void fasp_precond_ilu (
            REAL * r,
            REAL * z,
            void * data )
```

ILU preconditioner.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Shiquan Zhang

Date

04/06/2010

Definition at line 198 of file PreCSR.c.

9.76.2.8 fasp_precond_ilu_backward()

```
void fasp_precond_ilu_backward (
            REAL * r,
            REAL * z,
            void * data )
```

ILU preconditioner: only backward sweep.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu, Shiquan Zhang

Date

04/06/2010

Definition at line 315 of file PreCSR.c.

9.76.2.9 fasp_precond_ilu_forward()

```
void fasp_precond_ilu_forward (
            REAL * r,
            REAL * z,
            void * data )
```

ILU preconditioner: only forward sweep.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu, Shiquang Zhang

Date

04/06/2010

Definition at line 262 of file PreCSR.c.

9.76.2.10 fasp_precond_namli()

```
void fasp_precond_namli (
            REAL * r,
            REAL * z,
            void * data )
```

Nonlinear AMLI AMG preconditioner.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

04/25/2011

Definition at line 512 of file PreCSR.c.

9.76.2.11 fasp_precond_setup()

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

Setup preconditioner interface for iterative methods.

Parameters

precond_type	Preconditioner type
amgparam	Pointer to AMG parameters
iluparam	Pointer to ILU parameters
A	Pointer to the coefficient matrix

Returns

Pointer to preconditioner

Author

Feiteng Huang

Date

05/18/2009

Definition at line 46 of file PreCSR.c.

9.76.2.12 fasp_precond_swz()

```
void fasp_precond_swz (
            REAL * r,
            REAL * z,
            void * data )
```

get z from r by Schwarz

Parameters

r	Pointer to residual
z	Pointer to preconditioned residual
data	Pointer to precondition data

Author

Xiaozhe Hu

Date

03/22/2010

Note

Change Schwarz interface by Zheng Li on 11/18/2014

Definition at line 368 of file PreCSR.c.

9.77 PreDataInit.c File Reference

Initialize important data structures.

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

Functions

- void fasp_precond_data_init (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.
- void fasp_amg_data_free (AMG_data ∗mgl, AMG_param ∗param)

  Free AMG_data data memeory space.
- AMG_data_bsr ∗ fasp_amg_data_bsr_create (SHORT max_levels)

  Create and initialize AMG_data data sturcture for AMG/SAMG (BSR format)
- void fasp_amg_data_bsr_free (AMG_data_bsr ∗mgl)

  Free AMG_data_bsr data memeory space.
- void fasp_ilu_data_create (const INT iwk, const INT nwork, ILU_data ∗iludata)

  Allocate workspace for ILU factorization.
- void fasp_ilu_data_free (ILU_data ∗ILUdata)

  Create ILU_data sturcture.
- void fasp_swz_data_free (SWZ_data ∗swzdata)

  Free SWZ_data data memeory space.

9.77.1 Detailed Description

Initialize important data structures.

Note

This file contains Level-4 (Pre) functions. It requires: AuxMemory.c, AuxVector.c, BlaSparseBSR.c, and Bla←↩ SparseCSR.c

Warning

Every structures should be initialized before usage.

9.77.2 Function Documentation

9.77.2.1 fasp_amg_data_bsr_create()

```
AMG_data_bsr * fasp_amg_data_bsr_create (
            SHORT max_levels )
```

Create and initialize AMG_data data sturcture for AMG/SAMG (BSR format)

Parameters

max_levels	Max number of levels allowed

Returns

Pointer to the AMG_data data structure

Author

Xiaozhe Hu

Date

08/07/2011

Definition at line 170 of file PreDataInit.c.

9.77.2.2 fasp_amg_data_bsr_free()

```
void fasp_amg_data_bsr_free (
            AMG_data_bsr * mgl )
```

Free AMG_data_bsr data memeory space.

Parameters

mgl	Pointer to the AMG_data_bsr

Author

Xiaozhe Hu

Date

2013/02/13

Definition at line 200 of file PreDataInit.c.

9.77.2.3 fasp_amg_data_create()

```
AMG_data * fasp_amg_data_create (
            SHORT max_levels )
```

Create and initialize AMG_data for classical and SA AMG.

Parameters

max_levels	Max number of levels allowed

Returns

Pointer to the AMG_data data structure

Author

Chensong Zhang

Date

2010/04/06

Definition at line 64 of file PreDataInit.c.

9.77.2.4 fasp_amg_data_free()

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

Free AMG_data data memeory space.

Parameters

mgl	Pointer to the AMG_data
param	Pointer to AMG parameters

Author

Chensong Zhang

Date

2010/04/06

Modified by Chensong Zhang on 05/05/2013: Clean up param as well! Modified by Hongxuan Zhang on 12/15/2015: free memory for Intel MKL PARDISO.

Definition at line 96 of file PreDataInit.c.

9.77.2.5 fasp_ilu_data_create()

```
void fasp_ilu_data_create (
            const INT iwk,
            const INT nwork,
            ILU_data * iludata )
```

Allocate workspace for ILU factorization.

Parameters

iwk	Size of the index array
nwork	Size of the work array
iludata	Pointer to the ILU_data

Author

Chensong Zhang

Date

2010/04/06

Definition at line 246 of file PreDataInit.c.

9.77.2.6 fasp_ilu_data_free()

```
void fasp_ilu_data_free (
            ILU_data * ILUdata )
```

Create ILU_data sturcture.

Parameters

ILUdata	Pointer to ILU_data

Author

Chensong Zhang

Date

2010/04/03

Definition at line 276 of file PreDataInit.c.

9.77.2.7 fasp_precond_data_init()

```
void fasp_precond_data_init (
            precond_data * pcdata )
```

Initialize precond_data.

Parameters

pcdata	Preconditioning data structure

Author

Chensong Zhang

Date

2010/03/23

Definition at line 33 of file PreDataInit.c.

9.77.2.8 fasp_swz_data_free()

```
void fasp_swz_data_free (
            SWZ_data * swzdata )
```

Free SWZ_data data memeory space.

Parameters

swzdata	Pointer to the SWZ_data for Schwarz methods

Author

Xiaozhe Hu

Date

2010/04/06

Definition at line 301 of file PreDataInit.c.

9.78 PreMGCycle.c File Reference

Abstract multigrid cycle – non-recursive version.

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

9.78.1 Detailed Description

Abstract multigrid cycle – non-recursive version.

Note

This file contains Level-4 (Pre) functions. It requires: AuxArray.c, AuxMessage.c, AuxVector.c, BlaArray.c, Bla←SchwarzSetup.c, BlaSpmvBSR.c, BlaSpmvCSR.c, ItrSmootherBSR.c, ItrSmootherCSR.c, ItrSmootherCSRpoly.c, KryPcg.c, KryPvgmres.c, KrySPcg.c, and KrySPvgmres.c

9.78.2 Function Documentation

9.78.2.1 fasp_solver_mgcycle()

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

Solve Ax=b with non-recursive multigrid cycle.

Parameters

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

Author

Chensong Zhang

Date

10/06/2010

Modified by Chensong Zhang on 02/27/2013: update direct solvers. Modified by Chensong Zhang on 12/30/2014: update Schwarz smoothers.

Definition at line 56 of file PreMGCycle.c.

9.78.2.2 fasp_solver_mgcycle_bsr()

```
void fasp_solver_mgcycle_bsr (
            AMG_data_bsr * mgl,
            AMG_param * param )
```

Solve Ax=b with non-recursive multigrid cycle.

Parameters

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

Author

Xiaozhe Hu

Date

08/07/2011

Definition at line 280 of file PreMGCycle.c.

9.79 PreMGCycleFull.c File Reference

Abstract non-recursive full multigrid cycle.

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

Functions

- void fasp_solver_fmgcycle (AMG_data *mgl, AMG_param *param)

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

9.79.1 Detailed Description

Abstract non-recursive full multigrid cycle.

Note

This file contains Level-4 (Pre) functions. It requires: AuxArray.c, AuxMessage.c, AuxVector.c, BlaSchwarz↩ Setup.c, BlaArray.c, BlaSpmvCSR.c, BlaVector.c, ItrSmootherCSR.c, ItrSmootherCSRpoly.c, KryPcg.c, KryS↩ Pcg.c, and KrySPvgmres.c

9.79.2 Function Documentation

9.79.2.1 fasp_solver_fmgcycle()

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

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

Parameters

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

Author

Chensong Zhang

Date

02/27/2011

Modified by Chensong Zhang on 06/01/2012: fix a bug when there is only one level. Modified by Hongxuan Zhang on 12/15/2015: update direct solvers.

Definition at line 47 of file PreMGCycleFull.c.

9.80 PreMGRecur.c File Reference

Abstract multigrid cycle – recursive version.

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

Functions

- void fasp_solver_mgrecur (AMG_data ∗mgl, AMG_param ∗param, INT level)

  Solve Ax=b with recursive multigrid K-cycle.

9.80.1 Detailed Description

Abstract multigrid cycle – recursive version.

Note

This file contains Level-4 (Pre) functions. It requires: AuxArray.c, AuxMessage.c, AuxVector.c, BlaSpmvCSR.c, ItrSmootherCSR.c, ItrSmootherCSRpoly.c, KryPcg.c, KrySPcg.c, and KrySPvgmres.c

TODO: Not used any more. Will be removed! –Chensong

9.80.2 Function Documentation

9.80.2.1 fasp_solver_mgrecur()

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

Solve Ax=b with recursive multigrid K-cycle.

Parameters

mgl	Pointer to AMG data: AMG_data
param	Pointer to AMG parameters: AMG_param
level	Index of the current level

Author

Xuehai Huang, Chensong Zhang

Date

04/06/2010

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

Definition at line 47 of file PreMGRecur.c.

9.81 PreMGRecurAMLI.c File Reference

Abstract AMLI multilevel iteration – recursive version.

```
#include <math.h>
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "PreMGUtil.inl"
#include "PreMGSmoother.inl"
#include "PreMGRecurAMLI.inl"
```

Functions

- void fasp_solver_amli (AMG_data ∗mgl, AMG_param ∗param, INT level)

  Solve Ax=b with recursive AMLI-cycle.
- void fasp_solver_namli (AMG_data ∗mgl, AMG_param ∗param, INT level, INT num_levels)

  Solve Ax=b with recursive nonlinear AMLI-cycle.
- void fasp_solver_namli_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.

9.81.1 Detailed Description

Abstract AMLI multilevel iteration – recursive version.

Note

This file contains Level-4 (Pre) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, AuxParam.c, AuxVector.c, BlaSchwarzSetup.c, BlaArray.c, BlaSpmvBSR.c, BlaSpmvCSR.c, ItrSmootherBSR.c, ItrSmootherCSR.c, ItrSmootherCSRpoly.c, KryPcg.c, KryPvfgmres.c, KrySPcg.c, KrySPvgmres.c, PreBSR.c, and PreCSR.c
This file includes both AMLI and non-linear AMLI cycles

9.81.2 Function Documentation

9.81.2.1 fasp_amg_amli_coef()

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

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

Parameters

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

Author

Xiaozhe Hu

Date

01/23/2011

Definition at line 719 of file PreMGRecurAMLI.c.

9.81.2.2 fasp_solver_amli()

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

Solve Ax=b with recursive AMLI-cycle.

Parameters

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

Author

Xiaozhe Hu

Date

01/23/2011

Note

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

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

Definition at line 58 of file PreMGRecurAMLI.c.

9.81.2.3 fasp_solver_namli()

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

Solve Ax=b with recursive nonlinear AMLI-cycle.

Parameters

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

Author

Xiaozhe Hu

Date

04/06/2010

Note

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

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

Definition at line 282 of file PreMGRecurAMLI.c.

9.81.2.4 fasp_solver_namli_bsr()

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

Solve Ax=b with recursive nonlinear AMLI-cycle.

Parameters

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

Author

Xiaozhe Hu

Date

04/06/2010

Note

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

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

Definition at line 521 of file PreMGRecurAMLI.c.

9.82 PreMGSolve.c File Reference

Algebraic multigrid iterations: SOLVE phase.

```
#include <time.h>
#include "fasp.h"
#include "fasp_functs.h"
#include "KryUtil.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_namli (AMG_data ∗mgl, AMG_param ∗param)
  Nonlinear AMLI – SOLVE phase.
- void fasp_famg_solve (AMG_data ∗mgl, AMG_param ∗param)
  FMG – SOLVE phase.

9.82.1 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!
This file contains Level-4 (Pre) functions. It requires: AuxMessage.c, AuxTiming.c, AuxVector.c, BlaSpmvCSR.c, BlaVector.c, PreMGCycle.c, PreMGCycleFull.c, and PreMGRecurAMLI.c

9.82.2 Function Documentation

9.82.2.1 fasp_amg_solve()

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

AMG – SOLVE phase.

Parameters

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

Returns

Iteration number if converges; ERROR otherwise.

Author

Xuehai Huang, Chensong Zhang

Date

04/02/2010

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

Definition at line 49 of file PreMGSolve.c.

9.82.2.2 fasp_amg_solve_amli()

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

AMLI – SOLVE phase.

Parameters

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

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

01/23/2011

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

Note

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

Definition at line 137 of file PreMGSolve.c.

9.82.2.3 fasp_amg_solve_namli()

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

Nonlinear AMLI – SOLVE phase.

Parameters

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

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

04/30/2011

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

Note

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

Definition at line 220 of file PreMGSolve.c.

9.82.2.4 fasp_famg_solve()

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

FMG – SOLVE phase.

Parameters

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

Author

Chensong Zhang

Date

01/10/2012

Definition at line 292 of file PreMGSolve.c.

9.83 PreSTR.c File Reference

Preconditioners for dSTRmat matrices.

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

Functions

- void fasp_precond_dstr_diag (REAL *r, REAL *z, void *data)
  *Diagonal preconditioner z=inv(D)*r.*
- void fasp_precond_dstr_ilu0 (REAL *r, REAL *z, void *data)
  Preconditioning using STR_ILU(0) decomposition.
- void fasp_precond_dstr_ilu1 (REAL *r, REAL *z, void *data)
  Preconditioning using STR_ILU(1) decomposition.
- void fasp_precond_dstr_ilu0_forward (REAL *r, REAL *z, void *data)
  Preconditioning using STR_ILU(0) decomposition: Lz = r.
- void fasp_precond_dstr_ilu0_backward (REAL *r, REAL *z, void *data)
  Preconditioning using STR_ILU(0) decomposition: Uz = r.
- void fasp_precond_dstr_ilu1_forward (REAL *r, REAL *z, void *data)
  Preconditioning using STR_ILU(1) decomposition: Lz = r.
- void fasp_precond_dstr_ilu1_backward (REAL *r, REAL *z, void *data)
  Preconditioning using STR_ILU(1) decomposition: Uz = r.
- void fasp_precond_dstr_blockgs (REAL *r, REAL *z, void *data)
  CPR-type preconditioner (STR format)

9.83.1 Detailed Description

Preconditioners for dSTRmat matrices.

Note

This file contains Level-4 (Pre) functions. It requires: AuxArray.c, AuxMemory.c, AuxVector.c, BlaSmallMat.c, BlaArray.c, and ItrSmootherSTR.c

9.83.2 Function Documentation

9.83.2.1 fasp_precond_dstr_blockgs()

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

CPR-type preconditioner (STR format)

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Shiquan Zhang

Date

10/17/2010

Definition at line 1723 of file PreSTR.c.

9.83.2.2 fasp_precond_dstr_diag()

```
void fasp_precond_dstr_diag (
            REAL * r,
            REAL * z,
            void * data )
```

Diagonal preconditioner z=inv(D)∗r.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Shiquan Zhang

Date

04/06/2010

Definition at line 44 of file PreSTR.c.

9.83.2.3 fasp_precond_dstr_ilu0()

```
void fasp_precond_dstr_ilu0 (
            REAL * r,
            REAL * z,
            void * data )
```

Preconditioning using STR_ILU(0) decomposition.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Shiquan Zhang

Date

04/21/2010

Definition at line 71 of file PreSTR.c.

9.83.2.4 fasp_precond_dstr_ilu0_backward()

```
void fasp_precond_dstr_ilu0_backward (
            REAL * r,
            REAL * z,
            void * data )
```

Preconditioning using STR_ILU(0) decomposition: Uz = r.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Shiquan Zhang

Date

06/07/2010

Definition at line 995 of file PreSTR.c.

9.83.2.5 fasp_precond_dstr_ilu0_forward()

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

Preconditioning using STR_ILU(0) decomposition: Lz = r.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Shiquan Zhang

Date

06/07/2010

Definition at line 832 of file PreSTR.c.

9.83.2.6 fasp_precond_dstr_ilu1()

```
void fasp_precond_dstr_ilu1 (
            REAL * r,
            REAL * z,
            void * data )
```

Preconditioning using STR_ILU(1) decomposition.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Shiquan Zhang

Date

04/21/2010

Definition at line 353 of file PreSTR.c.

9.83.2.7 fasp_precond_dstr_ilu1_backward()

```
void fasp_precond_dstr_ilu1_backward (
            REAL * r,
            REAL * z,
            void * data )
```

Preconditioning using STR_ILU(1) decomposition: Uz = r.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Shiquan Zhang

Date

04/21/2010

Definition at line 1442 of file PreSTR.c.

9.83.2.8 fasp_precond_dstr_ilu1_forward()

```
void fasp_precond_dstr_ilu1_forward (
            REAL * r,
            REAL * z,
            void * data )
```

Preconditioning using STR_ILU(1) decomposition: Lz = r.

Parameters

r	Pointer to the vector needs preconditioning
z	Pointer to preconditioned vector
data	Pointer to precondition data

Author

Shiquan Zhang

Date

04/21/2010

Definition at line 1176 of file PreSTR.c.

9.84 SolAMG.c File Reference

AMG method as an iterative solver.

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

Functions

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

  Solve Ax = b by algebraic multigrid methods.

9.84.1 Detailed Description

AMG method as an iterative solver.

Note

This file contains Level-5 (Sol) functions. It requires: AuxMessage.c, AuxTiming.c, AuxVector.c, BlaSparseCheck.c, BlaSparseCSR.c, KrySPgmres.c, PreAMGSetupRS.c, PreAMGSetupSA.c, PreAMGSetupUA.c, PreDataInit.c, and PreMGSolve.c

9.84.2 Function Documentation

9.84.2.1 fasp_solver_amg()

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

Solve Ax = b by algebraic multigrid methods.

Parameters

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

Author

Chensong Zhang

Date

04/06/2010

Note

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

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

Definition at line 46 of file SolAMG.c.

9.85 SolBLC.c File Reference

Iterative solvers for dBLCmat matrices.

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

Functions

- INT fasp_solver_dblc_itsolver (dBLCmat *A, dvector *b, dvector *x, precond *pc, ITS_param *itparam)

  Solve Ax = b by standard Krylov methods.
- INT fasp_solver_dblc_krylov (dBLCmat *A, dvector *b, dvector *x, ITS_param *itparam)

  Solve Ax = b by standard Krylov methods.
- INT fasp_solver_dblc_krylov_block_3 (dBLCmat *A, dvector *b, dvector *x, ITS_param *itparam, AMG_param *amgparam, dCSRmat *A_diag)

  Solve Ax = b by standard Krylov methods.
- INT fasp_solver_dblc_krylov_block_4 (dBLCmat *A, dvector *b, dvector *x, ITS_param *itparam, AMG_param *amgparam, dCSRmat *A_diag)

  Solve Ax = b by standard Krylov methods.
- INT fasp_solver_dblc_krylov_sweeping (dBLCmat *A, dvector *b, dvector *x, ITS_param *itparam, INT NumLayers, dBLCmat *Ai, dCSRmat *local_A, ivector *local_index)

  Solve Ax = b by standard Krylov methods.

9.85.1 Detailed Description

Iterative solvers for dBLCmat matrices.

Note

This file contains Level-5 (Sol) functions. It requires: AuxMemory.c, AuxMessage.c, AuxTiming.c, AuxVector.c, BlaSparseCSR.c, KryPbcgs.c, KryPgmres.c, KryPminres.c, KryPvbcgs.c, KryPvfgmres.c, KryPvgmres.c, PreAMGSetupRS.c, PreAMGSetupSA.c, PreAMGSetupUA.c, PreBLC.c, and PreDataInit.c

9.85.2 Function Documentation

9.85.2.1 fasp_solver_dblc_itsolver()

```
INT fasp_solver_dblc_itsolver (
            dBLCmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            ITS_param * itparam )
```

Solve Ax = b by standard Krylov methods.

Parameters

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

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

11/25/2010

Modified by Chunsheng Feng on 03/04/2016: add VBiCGstab solver

Definition at line 54 of file SolBLC.c.

9.85.2.2 fasp_solver_dblc_krylov()

```
INT fasp_solver_dblc_krylov (
            dBLCmat * A,
            dvector * b,
            dvector * x,
            ITS_param * itparam )
```

Solve Ax = b by standard Krylov methods.

Parameters

A	Pointer to the coeff matrix in dBLCmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

07/18/2010

Definition at line 146 of file SolBLC.c.

9.85.2.3 fasp_solver_dblc_krylov_block_3()

```
INT fasp_solver_dblc_krylov_block_3 (
            dBLCmat * A,
            dvector * b,
            dvector * x,
            ITS_param * itparam,
            AMG_param * amgparam,
            dCSRmat * A_diag )
```

Solve Ax = b by standard Krylov methods.

Parameters

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

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

07/10/2014

Warning

Only works for 3by3 block dCSRmat problems!! – Xiaozhe Hu

Definition at line 200 of file SolBLC.c.

9.85.2.4 fasp_solver_dblc_krylov_block_4()

```
INT fasp_solver_dblc_krylov_block_4 (
            dBLCmat * A,
            dvector * b,
            dvector * x,
            ITS_param * itparam,
            AMG_param * amgparam,
            dCSRmat * A_diag )
```

Solve Ax = b by standard Krylov methods.

Parameters

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

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

07/06/2014

Warning

Only works for 4 by 4 block dCSRmat problems!! – Xiaozhe Hu

Definition at line 397 of file SolBLC.c.

9.85.2.5 fasp_solver_dblc_krylov_sweeping()

```
INT fasp_solver_dblc_krylov_sweeping (
            dBLCmat * A,
            dvector * b,
            dvector * x,
            ITS_param * itparam,
            INT NumLayers,
            dBLCmat * Ai,
            dCSRmat * local_A,
            ivector * local_index )
```

Solve Ax = b by standard Krylov methods.

Parameters

A	Pointer to the coeff matrix in dBLCmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers
NumLayers	Number of layers used for sweeping preconditioner
Ai	Pointer to the coeff matrix for the preconditioner in dBLCmat format

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

05/01/2014

Definition at line 523 of file SolBLC.c.

9.86 SolBSR.c File Reference

Iterative solvers for dBSRmat matrices.

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

Functions

- INT fasp_solver_dbsr_itsolver (dBSRmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, ITS_param ∗itparam)

  Solve Ax=b by preconditioned Krylov methods for BSR matrices.
- INT fasp_solver_dbsr_krylov (dBSRmat ∗A, dvector ∗b, dvector ∗x, ITS_param ∗itparam)

  Solve Ax=b by standard Krylov methods for BSR matrices.
- INT fasp_solver_dbsr_krylov_diag (dBSRmat ∗A, dvector ∗b, dvector ∗x, ITS_param ∗itparam)

  Solve Ax=b by diagonal preconditioned Krylov methods.
- INT fasp_solver_dbsr_krylov_ilu (dBSRmat ∗A, dvector ∗b, dvector ∗x, ITS_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, ITS_param ∗itparam, AMG_param ∗amgparam)

  Solve Ax=b by AMG preconditioned Krylov methods.
- INT fasp_solver_dbsr_krylov_amg_nk (dBSRmat ∗A, dvector ∗b, dvector ∗x, ITS_param ∗itparam, AMG_param ∗amgparam, dCSRmat ∗A_nk, dCSRmat ∗P_nk, dCSRmat ∗R_nk)

  Solve Ax=b by AMG with extra near kernel solve preconditioned Krylov methods.
- INT fasp_solver_dbsr_krylov_nk_amg (dBSRmat ∗A, dvector ∗b, dvector ∗x, ITS_param ∗itparam, AMG_param ∗amgparam, const INT nk_dim, dvector ∗nk)

  Solve Ax=b by AMG preconditioned Krylov methods with extra kernal space.

9.86.1 Detailed Description

Iterative solvers for dBSRmat matrices.

Note

This file contains Level-5 (Sol) functions. It requires: AuxMemory.c, AuxMessage.c, AuxThreads.c, AuxTiming.c, AuxVector.c, BlaSmallMatInv.c, BlaILUSetupBSR.c, BlaSparseBSR.c, BlaSparseCheck.c, KryPbcgs.c, KryPcg.c, KryPgmres.c, KryPvbcgs.c, KryPvfgmres.c, KryPvgmres.c, PreAMGSetupSA.c, PreAMGSetupUA.c, PreBSR.c, and PreDataInit.c

9.86.2 Function Documentation

9.86.2.1 fasp_solver_dbsr_itsolver()

```
INT fasp_solver_dbsr_itsolver (
            dBSRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            ITS_param * itparam )
```

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

Parameters

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

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou, Xiaozhe Hu

Date

10/26/2010

Modified by Chunsheng Feng on 03/04/2016: add VBiCGstab solver

Definition at line 56 of file SolBSR.c.

9.86.2.2 fasp_solver_dbsr_krylov()

```
INT fasp_solver_dbsr_krylov (
            dBSRmat * A,
            dvector * b,
            dvector * x,
            ITS_param * itparam )
```

Solve Ax=b by standard Krylov methods for BSR matrices.

Parameters

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

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou, Xiaozhe Hu

Date

10/26/2010

Definition at line 149 of file SolBSR.c.

9.86.2.3 fasp_solver_dbsr_krylov_amg()

```
INT fasp_solver_dbsr_krylov_amg (
            dBSRmat * A,
            dvector * b,
            dvector * x,
            ITS_param * itparam,
            AMG_param * amgparam )
```

Solve Ax=b by AMG preconditioned Krylov methods.

Parameters

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

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

03/16/2012

parameters of iterative method

Definition at line 370 of file SolBSR.c.

9.86.2.4 fasp_solver_dbsr_krylov_amg_nk()

```
INT fasp_solver_dbsr_krylov_amg_nk (
            dBSRmat * A,
            dvector * b,
            dvector * x,
            ITS_param * itparam,
            AMG_param * amgparam,
            dCSRmat * A_nk,
            dCSRmat * P_nk,
            dCSRmat * R_nk )
```

Solve Ax=b by AMG with extra near kernel solve preconditioned Krylov methods.

Parameters

A	Pointer to the coeff matrix in dBSRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers
amgparam	Pointer to parameters of AMG
A_nk	Pointer to the coeff matrix for near kernel space in dBSRmat format
P_nk	Pointer to the prolongation for near kernel space in dBSRmat format
R_nk	Pointer to the restriction for near kernel space in dBSRmat format

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

05/26/2012

Definition at line 512 of file SolBSR.c.

9.86.2.5 fasp_solver_dbsr_krylov_diag()

```
INT fasp_solver_dbsr_krylov_diag (
            dBSRmat * A,
            dvector * b,
            dvector * x,
            ITS_param * itparam )
```

Solve Ax=b by diagonal preconditioned Krylov methods.

Parameters

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

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou, Xiaozhe Hu

Date

10/26/2010

Modified by Chunsheng Feng, Zheng Li on 10/15/2012

Definition at line 199 of file SolBSR.c.

9.86.2.6 fasp_solver_dbsr_krylov_ilu()

```
INT fasp_solver_dbsr_krylov_ilu (
            dBSRmat * A,
            dvector * b,
            dvector * x,
            ITS_param * itparam,
            ILU_param * iluparam )
```

Solve Ax=b by ILUs preconditioned Krylov methods.

Parameters

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

Returns

Iteration number if converges; ERROR otherwise.

Author

Shiquang Zhang, Xiaozhe Hu

Date

10/26/2010

Definition at line 303 of file SolBSR.c.

9.86.2.7 fasp_solver_dbsr_krylov_nk_amg()

```
INT fasp_solver_dbsr_krylov_nk_amg (
            dBSRmat * A,
            dvector * b,
            dvector * x,
            ITS_param * itparam,
            AMG_param * amgparam,
            const INT nk_dim,
            dvector * nk )
```

Solve Ax=b by AMG preconditioned Krylov methods with extra kernal space.

Parameters

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

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

05/27/2012

parameters of iterative method

Definition at line 671 of file SolBSR.c.

9.87 SolCSR.c File Reference

Iterative solvers for dCSRmat matrices.

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

Functions

- INT fasp_solver_dcsr_itsolver (dCSRmat *A, dvector *b, dvector *x, precond *pc, ITS_param *itparam)

  Solve Ax=b by preconditioned Krylov methods for CSR matrices.
- INT fasp_solver_dcsr_krylov (dCSRmat *A, dvector *b, dvector *x, ITS_param *itparam)

  Solve Ax=b by standard Krylov methods for CSR matrices.
- INT fasp_solver_dcsr_krylov_diag (dCSRmat *A, dvector *b, dvector *x, ITS_param *itparam)

  Solve Ax=b by diagonal preconditioned Krylov methods.

- INT fasp_solver_dcsr_krylov_swz (dCSRmat ∗A, dvector ∗b, dvector ∗x, ITS_param ∗itparam, SWZ_param ∗schparam)

  Solve Ax=b by overlapping Schwarz Krylov methods.
- INT fasp_solver_dcsr_krylov_amg (dCSRmat ∗A, dvector ∗b, dvector ∗x, ITS_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, ITS_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, ITS_param ∗itparam, ILU_param ∗iluparam, dCSRmat ∗M)

  Solve Ax=b by ILUs preconditioned Krylov methods: ILU of M as preconditioner.
- INT fasp_solver_dcsr_krylov_amg_nk (dCSRmat ∗A, dvector ∗b, dvector ∗x, ITS_param ∗itparam, AMG_param ∗amgparam, dCSRmat ∗A_nk, dCSRmat ∗P_nk, dCSRmat ∗R_nk)

  Solve Ax=b by AMG preconditioned Krylov methods with an extra near kernel solve.

9.87.1 Detailed Description

Iterative solvers for dCSRmat matrices.

Note

This file contains Level-5 (Sol) functions. It requires: AuxMemory.c, AuxMessage.c, AuxParam.c, AuxTiming.c, AuxVector.c, BlaILUSetupCSR.c, BlaSchwarzSetup.c, BlaSparseCheck.c, BlaSparseCSR.c, KryPbcgs.c, Kry←Pcg.c, KryPgcg.c, KryPgcr.c, KryPgmres.c, KryPminres.c, KryPvbcgs.c, KryPvfgmres.c, KryPvgmres.c, PreAM←GSetupRS.c, PreAMGSetupSA.c, PreAMGSetupUA.c, PreCSR.c, and PreDataInit.c

9.87.2 Function Documentation

9.87.2.1 fasp_solver_dcsr_itsolver()

```
INT fasp_solver_dcsr_itsolver (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            ITS_param * itparam )
```

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

Note

This is an abstract interface for iterative methods.

Parameters

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

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

09/25/2009

Modified by Chunsheng Feng on 03/04/2016: add VBiCGstab solver

Definition at line 58 of file SolCSR.c.

9.87.2.2 fasp_solver_dcsr_krylov()

```
INT fasp_solver_dcsr_krylov (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            ITS_param * itparam )
```

Solve Ax=b by standard Krylov methods for CSR matrices.

Parameters

A	Pointer to the coeff matrix in dCSRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang, Shiquan Zhang

Date

09/25/2009

Definition at line 170 of file SolCSR.c.

9.87.2.3 fasp_solver_dcsr_krylov_amg()

```
INT fasp_solver_dcsr_krylov_amg (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            ITS_param * itparam,
            AMG_param * amgparam )
```

Solve Ax=b by AMG preconditioned Krylov methods.

Parameters

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

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

09/25/2009

Definition at line 364 of file SolCSR.c.

9.87.2.4 fasp_solver_dcsr_krylov_amg_nk()

```
INT fasp_solver_dcsr_krylov_amg_nk (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            ITS_param * itparam,
            AMG_param * amgparam,
            dCSRmat * A_nk,
            dCSRmat * P_nk,
            dCSRmat * R_nk )
```

Solve Ax=b by AMG preconditioned Krylov methods with an extra near kernel solve.

Parameters

A	Pointer to the coeff matrix in dCSRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers
amgparam	Pointer to parameters for AMG methods
A_nk	Pointer to the coeff matrix of near kernel space in dCSRmat format
P_nk	Pointer to the prolongation of near kernel space in dCSRmat format
R_nk	Pointer to the restriction of near kernel space in dCSRmat format

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

05/26/2014

Definition at line 637 of file SolCSR.c.

9.87.2.5 fasp_solver_dcsr_krylov_diag()

```
INT fasp_solver_dcsr_krylov_diag (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            ITS_param * itparam )
```

Solve Ax=b by diagonal preconditioned Krylov methods.

Parameters

A	Pointer to the coeff matrix in dCSRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang, Shiquan Zhang

Date

09/25/2009

Definition at line 220 of file SolCSR.c.

9.87.2.6 fasp_solver_dcsr_krylov_ilu()

```
INT fasp_solver_dcsr_krylov_ilu (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            ITS_param * itparam,
            ILU_param * iluparam )
```

Solve Ax=b by ILUs preconditioned Krylov methods.

Parameters

A	Pointer to the coeff matrix in dCSRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers
iluparam	Pointer to parameters for ILU

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang, Shiquan Zhang

Date

09/25/2009

Definition at line 469 of file SolCSR.c.

9.87.2.7 fasp_solver_dcsr_krylov_ilu_M()

```
INT fasp_solver_dcsr_krylov_ilu_M (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            ITS_param * itparam,
            ILU_param * iluparam,
            dCSRmat * M )
```

Solve Ax=b by ILUs preconditioned Krylov methods: ILU of M as preconditioner.

Parameters

A	Pointer to the coeff matrix in dCSRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers
iluparam	Pointer to parameters for ILU
M	Pointer to the preconditioning matrix in dCSRmat format

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

09/25/2009

Note

This function is specially designed for reservoir simulation. Have not been tested in any other places.

Definition at line 553 of file SolCSR.c.

9.87.2.8 fasp_solver_dcsr_krylov_swz()

```
INT fasp_solver_dcsr_krylov_swz (
            dCSRmat * A,
            dvector * b,
            dvector * x,
            ITS_param * itparam,
            SWZ_param * schparam )
```

Solve Ax=b by overlapping Schwarz Krylov methods.

Parameters

A	Pointer to the coeff matrix in dCSRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers
schparam	Pointer to parameters for Schwarz methods

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

03/21/2011

Modified by Chensong on 07/02/2012: change interface

Definition at line 283 of file SolCSR.c.

9.88 SolFAMG.c File Reference

Full AMG method as an iterative solver.

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

Functions

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

    Solve Ax=b by full AMG.

9.88.1 Detailed Description

Full AMG method as an iterative solver.

Note

This file contains Level-5 (Sol) functions. It requires: AuxMessage.c, AuxTiming.c, AuxVector.c, BlaSparse←Check.c, BlaSparseCSR.c, PreAMGSetupRS.c, PreAMGSetupSA.c, PreAMGSetupUA.c, PreDataInit.c, and PreMGSolve.c

9.88.2 Function Documentation

9.88.2.1 fasp_solver_famg()

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

Solve Ax=b by full AMG.

Parameters

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

Author

Xiaozhe Hu

Date

02/27/2011

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

Definition at line 41 of file SolFAMG.c.

9.89 SolGMGPoisson.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 "PreGMG.inl"
```

Functions

- INT fasp_poisson_gmg1d (REAL *u, REAL *b, const INT nx, const INT maxlevel, const REAL rtol, const SHORT prtlvl)

  Solve Ax=b of Poisson 1D equation by Geometric Multigrid Method.
- INT fasp_poisson_gmg2d (REAL *u, REAL *b, const INT nx, const INT ny, const INT maxlevel, const REAL rtol, const SHORT prtlvl)

  Solve Ax=b of Poisson 2D equation by Geometric Multigrid Method.
- INT fasp_poisson_gmg3d (REAL *u, REAL *b, const INT nx, const INT ny, const INT nz, const INT maxlevel, const REAL rtol, const SHORT prtlvl)

  Solve Ax=b of Poisson 3D equation by Geometric Multigrid Method.
- void fasp_poisson_fgmg1d (REAL *u, REAL *b, const INT nx, const INT maxlevel, const REAL rtol, const SHORT prtlvl)

  Solve Ax=b of Poisson 1D equation by Geometric Multigrid Method (FMG)
- void fasp_poisson_fgmg2d (REAL *u, REAL *b, const INT nx, const INT ny, const INT maxlevel, const REAL rtol, const SHORT prtlvl)

  Solve Ax=b of Poisson 2D equation by Geometric Multigrid Method (FMG)
- void fasp_poisson_fgmg3d (REAL *u, REAL *b, const INT nx, const INT ny, const INT nz, const INT maxlevel, const REAL rtol, const SHORT prtlvl)

  Solve Ax=b of Poisson 3D equation by Geometric Multigrid Method (FMG)
- INT fasp_poisson_gmgcg1d (REAL *u, REAL *b, const INT nx, const INT maxlevel, const REAL rtol, const SHORT prtlvl)

  Solve Ax=b of Poisson 1D equation by Geometric Multigrid Method (GMG preconditioned Conjugate Gradient method)
- INT fasp_poisson_gmgcg2d (REAL *u, REAL *b, const INT nx, const INT ny, const INT maxlevel, const REAL rtol, const SHORT prtlvl)

  Solve Ax=b of Poisson 2D equation by Geometric Multigrid Method (GMG preconditioned Conjugate Gradient method)
- INT fasp_poisson_gmgcg3d (REAL *u, REAL *b, const INT nx, const INT ny, const INT nz, const INT maxlevel, const REAL rtol, const SHORT prtlvl)

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

9.89.1 Detailed Description

GMG method as an iterative solver for Poisson Problem.

Note

This file contains Level-5 (Sol) functions. It requires: AuxArray.c, AuxMessage.c, and AuxTiming.c

9.89.2 Function Documentation

9.89.2.1 fasp_poisson_fgmg1d()

```
void fasp_poisson_fgmg1d (
            REAL * u,
            REAL * b,
            const INT nx,
            const INT maxlevel,
            const REAL rtol,
            const SHORT prtlvl )
```

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

Parameters

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

Author

Ziteng Wang, Chensong Zhang

Date

06/07/2013

Definition at line 442 of file SolGMGPoisson.c.

9.89.2.2 fasp_poisson_fgmg2d()

```
void fasp_poisson_fgmg2d (
            REAL * u,
            REAL * b,
            const INT nx,
            const INT ny,
            const INT maxlevel,
            const REAL rtol,
            const SHORT prtlvl )
```

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

Parameters

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

Author

Ziteng Wang, Chensong Zhang

Date

06/07/2013

Definition at line 535 of file SolGMGPoisson.c.

9.89.2.3 fasp_poisson_fgmg3d()

```
void fasp_poisson_fgmg3d (
            REAL * u,
            REAL * b,
            const INT nx,
            const INT ny,
            const INT nz,
            const INT maxlevel,
            const REAL rtol,
            const SHORT prtlvl )
```

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

Parameters

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

Author

Ziteng Wang, Chensong Zhang

Date

06/07/2013

Definition at line 642 of file SolGMGPoisson.c.

9.89.2.4 fasp_poisson_gmg1d()

```
INT fasp_poisson_gmg1d (
            REAL * u,
            REAL * b,
            const INT nx,
            const INT maxlevel,
            const REAL rtol,
            const SHORT prtlvl )
```

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

Parameters

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

Returns

Iteration number if converges; ERROR otherwise.

Author

Ziteng Wang, Chensong Zhang

Date

06/07/2013

Definition at line 48 of file SolGMGPoisson.c.

9.89.2.5 fasp_poisson_gmg2d()

```
INT fasp_poisson_gmg2d (
            REAL * u,
            REAL * b,
            const INT nx,
            const INT ny,
            const INT maxlevel,
            const REAL rtol,
            const SHORT prtlvl )
```

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

Parameters

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

Returns

Iteration number if converges; ERROR otherwise.

Author

Ziteng Wang, Chensong Zhang

Date

06/07/2013

Definition at line 172 of file SolGMGPoisson.c.

9.89.2.6 fasp_poisson_gmg3d()

```
INT fasp_poisson_gmg3d (
            REAL * u,
            REAL * b,
            const INT nx,
            const INT ny,
            const INT nz,
            const INT maxlevel,
            const REAL rtol,
            const SHORT prtlvl )
```

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

Parameters

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

Returns

Iteration number if converges; ERROR otherwise.

Author

Ziteng Wang, Chensong Zhang

Date

06/07/2013

Definition at line 308 of file SolGMGPoisson.c.

9.89.2.7 fasp_poisson_gmgcg1d()

```
INT fasp_poisson_gmgcg1d (
            REAL * u,
            REAL * b,
            const INT nx,
            const INT maxlevel,
            const REAL rtol,
            const SHORT prtlvl )
```

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

Parameters

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

Returns

Iteration number if converges; ERROR otherwise.

Author

Ziteng Wang, Chensong Zhang

Date

06/07/2013

Definition at line 751 of file SolGMGPoisson.c.

9.89.2.8 fasp_poisson_gmgcg2d()

```
INT fasp_poisson_gmgcg2d (
            REAL * u,
            REAL * b,
            const INT nx,
            const INT ny,
            const INT maxlevel,
            const REAL rtol,
            const SHORT prtlvl )
```

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

Parameters

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

Returns

Iteration number if converges; ERROR otherwise.

Author

Ziteng Wang, Chensong Zhang

Date

06/07/2013

Definition at line 845 of file SolGMGPoisson.c.

9.89.2.9 fasp_poisson_gmgcg3d()

```
INT fasp_poisson_gmgcg3d (
              REAL * u,
              REAL * b,
              const INT nx,
              const INT ny,
              const INT nz,
              const INT maxlevel,
              const REAL rtol,
              const SHORT prtlvl )
```

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

Parameters

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

Returns

Iteration number if converges; ERROR otherwise.

Author

Ziteng Wang, Chensong Zhang

Date

06/07/2013

Definition at line 954 of file SolGMGPoisson.c.

9.90 SolMatFree.c File Reference

Iterative solvers using MatFree spmv operations.

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

Functions

- INT fasp_solver_itsolver (mxv_matfree ∗mf, dvector ∗b, dvector ∗x, precond ∗pc, ITS_param ∗itparam)

  Solve Ax=b by preconditioned Krylov methods for CSR matrices.
- INT fasp_solver_krylov (mxv_matfree ∗mf, dvector ∗b, dvector ∗x, ITS_param ∗itparam)

  Solve Ax=b by standard Krylov methods – without preconditioner.
- void fasp_solver_matfree_init (INT matrix_format, mxv_matfree ∗mf, void ∗A)

  Initialize MatFree (or non-specified format) itsovlers.

9.90.1 Detailed Description

Iterative solvers using MatFree spmv operations.

Note

This file contains Level-5 (Sol) functions. It requires: AuxMessage.c, AuxTiming.c, BlaSpmvBLC.c, BlaSpmvBSR.c, BlaSpmvCSR.c, BlaSpmvCSRL.c, BlaSpmvSTR.c, KryPbcgs.c, KryPcg.c, KryPgcg.c, KryPgmres.c, KryPminres.c, KryPvfgmres.c, and KryPvgmres.c

9.90.2 Function Documentation

9.90.2.1 fasp_solver_itsolver()

```
INT fasp_solver_itsolver (
            mxv_matfree * mf,
            dvector * b,
            dvector * x,
            precond * pc,
            ITS_param * itparam )
```

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

Note

This is an abstract interface for iterative methods.

Parameters

mf	Pointer to mxv_matfree MatFree spmv operation
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
pc	Pointer to the preconditioning action
itparam	Pointer to parameters for iterative solvers

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

09/25/2009

Modified by Feiteng Huang on 09/19/2012: matrix free

Definition at line 58 of file SolMatFree.c.

9.90.2.2 fasp_solver_krylov()

```
INT fasp_solver_krylov (
            mxv_matfree * mf,
            dvector * b,
            dvector * x,
            ITS_param * itparam )
```

Solve Ax=b by standard Krylov methods – without preconditioner.

Parameters

mf	Pointer to mxv_matfree MatFree spmv operation
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers

Returns

Number of iterations if succeed

Author

Chensong Zhang, Shiquan Zhang

Date

09/25/2009

Modified by Feiteng Huang on 09/20/2012: matrix free

Definition at line 163 of file SolMatFree.c.

9.90.2.3 fasp_solver_matfree_init()

```
void fasp_solver_matfree_init (
            INT matrix_format,
            mxv_matfree * mf,
            void * A )
```

Initialize MatFree (or non-specified format) itsovlers.

Parameters

matrix_format	matrix format
mf	Pointer to mxv_matfree MatFree spmv operation
A	void pointer to the coefficient 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 210 of file SolMatFree.c.

9.91 SolSTR.c File Reference

Iterative solvers for dSTRmat matrices.

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

Functions

- INT fasp_solver_dstr_itsolver (dSTRmat ∗A, dvector ∗b, dvector ∗x, precond ∗pc, ITS_param ∗itparam)
  Solve Ax=b by standard Krylov methods.
- INT fasp_solver_dstr_krylov (dSTRmat ∗A, dvector ∗b, dvector ∗x, ITS_param ∗itparam)
  Solve Ax=b by standard Krylov methods.
- INT fasp_solver_dstr_krylov_diag (dSTRmat ∗A, dvector ∗b, dvector ∗x, ITS_param ∗itparam)
  Solve Ax=b by diagonal preconditioned Krylov methods.
- INT fasp_solver_dstr_krylov_ilu (dSTRmat ∗A, dvector ∗b, dvector ∗x, ITS_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, ITS_param ∗itparam, ivector ∗neigh, ivector ∗order)
  Solve Ax=b by diagonal preconditioned Krylov methods.

9.91.1 Detailed Description

Iterative solvers for dSTRmat matrices.

Note

This file contains Level-5 (Sol) functions. It requires: AuxArray.c, AuxMemory.c, AuxMessage.c, AuxTiming.c, AuxVector.c, BlaSmallMatInv.c, BlaILUSetupSTR.c, BlaSparseSTR.c, ItrSmootherSTR.c, KryPbcgs.c, KryPcg.c, KryPgmres.c, KryPvbcgs.c, KryPvgmres.c, and PreSTR.c

9.91.2 Function Documentation

9.91.2.1 fasp_solver_dstr_itsolver()

```
INT fasp_solver_dstr_itsolver (
            dSTRmat * A,
            dvector * b,
            dvector * x,
            precond * pc,
            ITS_param * itparam )
```

Solve Ax=b by standard Krylov methods.

Parameters

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

Returns

Iteration number if converges; ERROR otherwise.

Author

Chensong Zhang

Date

09/25/2009

Modified by Chunsheng Feng on 03/04/2016: add VBiCGstab solver

Definition at line 52 of file SolSTR.c.

9.91.2.2 fasp_solver_dstr_krylov()

```
INT fasp_solver_dstr_krylov (
            dSTRmat * A,
            dvector * b,
            dvector * x,
            ITS_param * itparam )
```

Solve Ax=b by standard Krylov methods.

Parameters

A	Pointer to the coeff matrix in dSTRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou

Date

04/25/2010

Definition at line 140 of file SolSTR.c.

9.91.2.3 fasp_solver_dstr_krylov_blockgs()

```
INT fasp_solver_dstr_krylov_blockgs (
            dSTRmat * A,
            dvector * b,
            dvector * x,
            ITS_param * itparam,
            ivector * neigh,
            ivector * order )
```

Solve Ax=b by diagonal preconditioned Krylov methods.

Parameters

A	Pointer to the coeff matrix in dSTRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers
neigh	Pointer to neighbor vector
order	Pointer to solver ordering

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

10/10/2010

Definition at line 347 of file SolSTR.c.

9.91.2.4 fasp_solver_dstr_krylov_diag()

```
INT fasp_solver_dstr_krylov_diag (
            dSTRmat * A,
            dvector * b,
            dvector * x,
            ITS_param * itparam )
```

Solve Ax=b by diagonal preconditioned Krylov methods.

Parameters

A	Pointer to the coeff matrix in dSTRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers

Returns

Iteration number if converges; ERROR otherwise.

Author

Zhiyang Zhou

Date

4/23/2010

Definition at line 188 of file SolSTR.c.

9.91.2.5 fasp_solver_dstr_krylov_ilu()

```
INT fasp_solver_dstr_krylov_ilu (
            dSTRmat * A,
            dvector * b,
            dvector * x,
            ITS_param * itparam,
            ILU_param * iluparam )
```

Solve Ax=b by structured ILU preconditioned Krylov methods.

Parameters

A	Pointer to the coeff matrix in dSTRmat format
b	Pointer to the right hand side in dvector format
x	Pointer to the approx solution in dvector format
itparam	Pointer to parameters for iterative solvers
iluparam	Pointer to parameters for ILU

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

05/01/2010

Definition at line 254 of file SolSTR.c.

9.92 SolWrapper.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)

9.92.1 Detailed Description

Wrappers for accessing functions by advanced users.

Note

This file contains Level-5 (Sol) functions. It requires: AuxParam.c, BlaFormat.c, BlaSparseBSR.c, BlaSparseC↩SR.c, SolAMG.c, SolBSR.c, and SolCSR.c

9.92.2 Function Documentation

9.92.2.1 fasp_fwrapper_amg_()

```
void fasp_fwrapper_amg_ (
            INT * n,
            INT * nnz,
            INT * ia,
            INT * ja,
            REAL * a,
            REAL * b,
            REAL * u,
            REAL * tol,
            INT * maxit,
            INT * ptrlvl )
```

Solve Ax=b by Ruge and Stuben's classic AMG.

Parameters

n	Number of cols of A
nnz	Number of nonzeros of A
ia	IA of A in CSR format
ja	JA of A in CSR format
a	VAL of A in CSR format
b	RHS vector
u	Solution vector
tol	Tolerance for iterative solvers
maxit	Max number of iterations
ptrlvl	Print level for iterative solvers

Author

Chensong Zhang

Date

09/16/2010

Definition at line 44 of file SolWrapper.c.

9.92.2.2 fasp_fwrapper_krylov_amg_()

```
void fasp_fwrapper_krylov_amg_ (
            INT * n,
            INT * nnz,
            INT * ia,
            INT * ja,
            REAL * a,
            REAL * b,
            REAL * u,
            REAL * tol,
            INT * maxit,
            INT * ptrlvl )
```

Solve Ax=b by Krylov method preconditioned by classic AMG.

Parameters

n	Number of cols of A
nnz	Number of nonzeros of A
ia	IA of A in CSR format
ja	JA of A in CSR format
a	VAL of A in CSR format
b	RHS vector
u	Solution vector
tol	Tolerance for iterative solvers
maxit	Max number of iterations
ptrlvl	Print level for iterative solvers

Author

Chensong Zhang

Date

09/16/2010

Definition at line 94 of file SolWrapper.c.

9.92.2.3 fasp_wrapper_dbsr_krylov_amg()

```
INT fasp_wrapper_dbsr_krylov_amg (
            INT n,
            INT nnz,
            INT nb,
            INT * ia,
            INT * ja,
            REAL * a,
            REAL * b,
            REAL * u,
            REAL tol,
            INT maxit,
            INT ptrlvl )
```

Solve Ax=b by Krylov method preconditioned by AMG (dcsr - > dbsr)

Parameters

n	Number of cols of A
nnz	Number of nonzeros of A
nb	Size of each small block
ia	IA of A in CSR format
ja	JA of A in CSR format
a	VAL of A in CSR format
b	RHS vector
u	Solution vector
tol	Tolerance for iterative solvers
maxit	Max number of iterations
ptrlvl	Print level for iterative solvers

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

03/05/2013

Definition at line 161 of file SolWrapper.c.

9.92.2.4 fasp_wrapper_dcoo_dbsr_krylov_amg()

```
INT fasp_wrapper_dcoo_dbsr_krylov_amg (
            INT n,
            INT nnz,
            INT nb,
            INT * ia,
            INT * ja,
            REAL * a,
            REAL * b,
            REAL * u,
            REAL tol,
            INT maxit,
            INT ptrlvl )
```

Solve Ax=b by Krylov method preconditioned by AMG (dcoo - $>$ dbsr)

Parameters

n	Number of cols of A
nnz	Number of nonzeros of A
nb	Size of each small block
ia	IA of A in COO format
ja	JA of A in COO format
a	VAL of A in COO format
b	RHS vector
u	Solution vector
tol	Tolerance for iterative solvers
maxit	Max number of iterations
ptrlvl	Print level for iterative solvers

Returns

Iteration number if converges; ERROR otherwise.

Author

Xiaozhe Hu

Date

03/06/2013

Definition at line 247 of file SolWrapper.c.

9.93 XtrMumps.c File Reference

Interface to MUMPS direct solvers.

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

Macros

- #define ICNTL(I) icntl[(I)-1]

Functions

- int fasp_solver_mumps (dCSRmat ∗ptrA, dvector ∗b, dvector ∗u, const SHORT prtlvl)

  Solve Ax=b by MUMPS directly.
- int fasp_solver_mumps_steps (dCSRmat ∗ptrA, dvector ∗b, dvector ∗u, Mumps_data ∗mumps)

  Solve Ax=b by MUMPS in three steps.

9.93.1 Detailed Description

Interface to MUMPS direct solvers.

Reference for MUMPS: `http://mumps.enseeiht.fr/`

9.93.2 Macro Definition Documentation

9.93.2.1 ICNTL

```
#define ICNTL(
              I ) icntl[(I)-1]
```

macro s.t. indices match documentation

Definition at line 23 of file XtrMumps.c.

9.93.3 Function Documentation

9.93.3.1 fasp_solver_mumps()

```
int fasp_solver_mumps (
             dCSRmat * ptrA,
             dvector * b,
             dvector * u,
             const SHORT prtlvl )
```

Solve Ax=b by MUMPS directly.

Parameters

ptrA	Pointer to a dCSRmat matrix
b	Pointer to the dvector of right-hand side term
u	Pointer to the dvector of solution
prtlvl	Output level

Author

Chunsheng Feng

Date

02/27/2013

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

Definition at line 45 of file XtrMumps.c.

9.93.3.2 fasp_solver_mumps_steps()

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

Solve Ax=b by MUMPS in three steps.

Parameters

ptrA	Pointer to a dCSRmat matrix
b	Pointer to the dvector of right-hand side term
u	Pointer to the dvector of solution
mumps	Pointer to MUMPS data

Author

Chunsheng Feng

Date

02/27/2013

Modified by Chensong Zhang on 02/27/2013 for new FASP function names. Modified by Zheng Li on 10/10/2014 to adjust input parameters.

Definition at line 175 of file XtrMumps.c.

9.94 XtrPardiso.c File Reference

Interface to Intel MKL PARDISO direct solvers.

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

Functions

- INT fasp_solver_pardiso (dCSRmat ∗ptrA, dvector ∗b, dvector ∗u, const SHORT prtlvl)

    Solve Ax=b by PARDISO directly. Each row of A should be in ascending order w.r.t. column indices.

9.94.1 Detailed Description

Interface to Intel MKL PARDISO direct solvers.

Reference for Intel MKL PARDISO: https://software.intel.com/en-us/node/470282

9.94.2 Function Documentation

9.94.2.1 fasp_solver_pardiso()

```
int fasp_solver_pardiso (
            dCSRmat * ptrA,
            dvector * b,
            dvector * u,
            const SHORT prtlvl )
```

Solve Ax=b by PARDISO directly. Each row of A should be in ascending order w.r.t. column indices.

Parameters

ptrA	Pointer to a dCSRmat matrix
b	Pointer to the dvector of right-hand side term
u	Pointer to the dvector of solution
prtlvl	Output level

Author

Hongxuan Zhang

Date

11/28/2015

Definition at line 44 of file XtrPardiso.c.

9.95 XtrSamg.c File Reference

Interface to SAMG solvers.

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

Functions

- void dvector2SAMGInput (dvector ∗vec, char ∗filename)

  Write a dvector to disk file in SAMG format (coordinate format)
- INT dCSRmat2SAMGInput (dCSRmat ∗A, char ∗filefrm, char ∗fileamg)

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

9.95.1 Detailed Description

Interface to SAMG solvers.

Reference for SAMG: http://www.scai.fraunhofer.de/geschaeftsfelder/nuso/produkte/samg.↩
html

Warning

This interface has *only* been tested for SAMG24a1 (2010 version)!

9.95.2 Function Documentation

9.95.2.1 dCSRmat2SAMGInput()

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

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

Parameters

A	Pointer to the dCSRmat matrix
filefrm	Name of the .frm file
fileamg	Name of the .amg file

Author

Zhiyang Zhou

Date

2010/08/25

Definition at line 65 of file XtrSamg.c.

9.95.2.2 dvector2SAMGInput()

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

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

Parameters

vec	Pointer to the dvector
filename	File name for input

Author

Zhiyang Zhou

Date

08/25/2010

Definition at line 36 of file XtrSamg.c.

9.96 XtrSuperlu.c File Reference

Interface to SuperLU direct solvers.

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

Functions

- int fasp_solver_superlu (dCSRmat ∗ptrA, dvector ∗b, dvector ∗u, const SHORT prtlvl)

  Solve Au=b by SuperLU.

9.96.1 Detailed Description

Interface to SuperLU direct solvers.

Reference for SuperLU: http://crd-legacy.lbl.gov/~xiaoye/SuperLU/

9.96.2 Function Documentation

9.96.2.1 fasp_solver_superlu()

```
int fasp_solver_superlu (
            dCSRmat * ptrA,
            dvector * b,
            dvector * u,
            const SHORT prtlvl )
```

Solve Au=b by SuperLU.

Parameters

ptrA	Pointer to a dCSRmat matrix
b	Pointer to the dvector of right-hand side term
u	Pointer to the dvector of solution
prtlvl	Output level

Author

Xiaozhe Hu

Date

11/05/2009

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

Definition at line 45 of file XtrSuperlu.c.

9.97 XtrUmfpack.c File Reference

Interface to UMFPACK direct solvers.

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

Functions

- INT fasp_solver_umfpack (dCSRmat *ptrA, dvector *b, dvector *u, const SHORT prtlvl)
  Solve Au=b by UMFpack.

9.97.1 Detailed Description

Interface to UMFPACK direct solvers.

Reference for SuiteSparse: http://faculty.cse.tamu.edu/davis/suitesparse.html

9.97.2 Function Documentation

9.97.2.1 fasp_solver_umfpack()

```
INT fasp_solver_umfpack (
            dCSRmat * ptrA,
            dvector * b,
            dvector * u,
            const SHORT prtlvl )
```

Solve Au=b by UMFpack.

Parameters

ptrA	Pointer to a dCSRmat matrix
b	Pointer to the dvector of right-hand side term
u	Pointer to the dvector of solution
prtlvl	Output level

Author

Chensong Zhang

Date

05/20/2010

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

Definition at line 43 of file XtrUmfpack.c.

Index