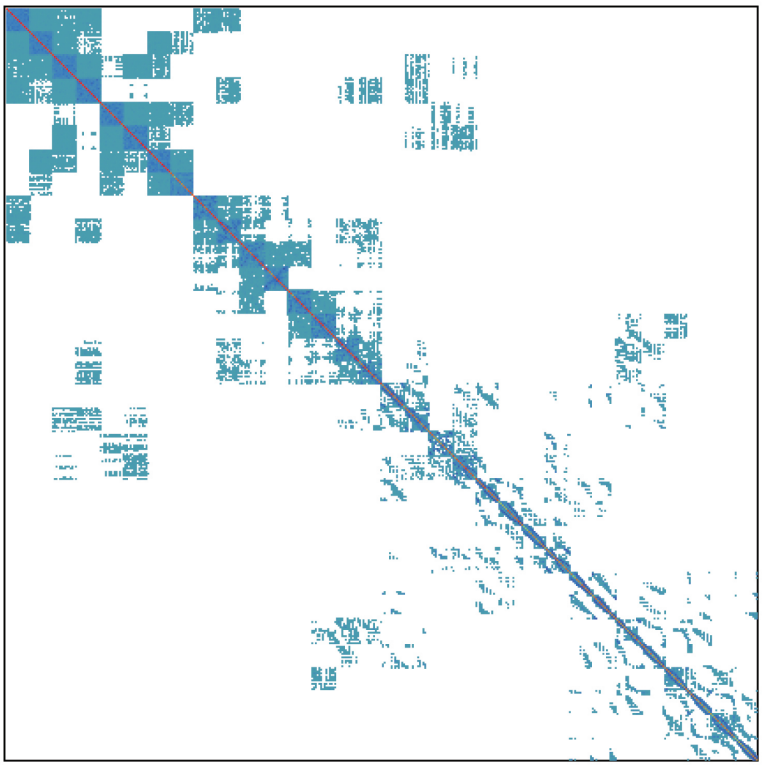
FSAIPACK Documentation

vers. 1.0



Generated by Doxygen 1.7.6.1

Jul 2013

Contents

1 Data Type Index **1**

1.1 Data Types List 1

2 File Index **3**

2.1 File List 3

3 Data Type Documentation **5**

3.1 class_CSRMAT::assignment(=) Interface Reference 5

3.1.1 Member Function/Subroutine Documentation 5

3.1.1.1 copy_CSRMAT 5

3.2 class_CSRMAT Module Reference 6

3.2.1 Detailed Description 6

3.2.2 Member Function/Subroutine Documentation 7

3.2.2.1 copy_CSRMAT 7

3.2.2.2 dlt_CSRCOEF 7

3.2.2.3 dlt_CSRMAT 8

3.2.2.4 errchk_CSRMAT 8

3.2.2.5 new_CSRCOEF 8

3.2.2.6 new_CSRMAT 9

3.3 class_FSAIP_Prec Module Reference 9

3.3.1 Detailed Description 10

3.3.2 Member Function/Subroutine Documentation 11

3.3.2.1 append_LEFT 11

3.3.2.2 append_RIGHT 11

3.3.2.3 apply_FSAIP_Prec 12

3.3.2.4 delete_LEFT 13

3.3.2.5 delete_RIGHT 13
3.3.2.6 dlt_FSAIP_Prec 14
3.3.2.7 getdata_FSAIP_Prec 14
3.3.2.8 wr_FSAIP_Prec 15
3.4 class_FSAIPACK Module Reference 16
3.4.1 Detailed Description 16
3.4.2 Member Function/Subroutine Documentation 17
3.4.2.1 CPT_adapt_FSAI 17
3.4.2.2 CPT_preconditioned_Matix 17
3.4.2.3 CPT_projection_FSAI 18
3.4.2.4 CPT_static_FSAI 19
3.4.2.5 ERRCHK_FSAIPACK 19
3.4.2.6 FILTER_FSAI 20
3.4.2.7 MK_patt 20
3.4.2.8 TRANSPOSE_FSAI 21
3.5 class_Pattern Module Reference 21
3.5.1 Detailed Description 22
3.5.2 Member Function/Subroutine Documentation 22
3.5.2.1 dlt_Pattern 22
3.5.2.2 new_Pattern 23
3.6 class_PCG_FSAIPACK Module Reference 23
3.6.1 Detailed Description 24
3.6.2 Member Function/Subroutine Documentation 24
3.6.2.1 errchk_PCG_FSAIPACK 24
3.6.2.2 set_PCG_FSAIPACK 25
3.6.2.3 solv_PCG_FSAIPACK 25
3.7 class_FSAIP_Prec::CSR_node Type Reference 26
3.7.1 Detailed Description 27
3.7.2 Member Data Documentation 27
3.7.2.1 next 27
3.7.2.2 pt_CSR 27
3.8 class_CSRMAT::CSRMAT Type Reference 27
3.8.1 Detailed Description 28
3.8.2 Member Data Documentation 28

3.8.2.1 coef 28
3.8.2.2 patt 28
3.9 class_FSAIP_Prec::FSAIP_Prec Type Reference 28
3.9.1 Detailed Description 29
3.9.2 Member Data Documentation 29
3.9.2.1 LEFT 29
3.9.2.2 RIGHT 29
3.10 class_Pattern::Pattern Type Reference 30
3.10.1 Detailed Description 30
3.10.2 Member Data Documentation 30
3.10.2.1 iat 30
3.10.2.2 ja 30
3.10.2.3 nrows 30
3.10.2.4 nterm 30
3.11 class_PCG_FSAIPACK::PCG_FSAIPACK Type Reference 31
3.11.1 Detailed Description 31
3.11.2 Member Data Documentation 31
3.11.2.1 bnorm 31
3.11.2.2 iout 31
3.11.2.3 isol 32
3.11.2.4 itmax 32
3.11.2.5 n_iter 32
3.11.2.6 resini 32
3.11.2.7 resiter 32
3.11.2.8 resreal 32
3.11.2.9 tol_CG 32

4 File Documentation **33**

4.1 A_PCG_FSAIPACK.f90 File Reference 33
4.2 A_precond.f90 File Reference 33
4.3 axbnsy.f90 File Reference 33
4.3.1 Function/Subroutine Documentation 34
4.3.1.1 axbnsy 34
4.4 B_CSRMAT.f90 File Reference 34

4.5 B_FSAIPACK.f90 File Reference 34
4.6 ddot_par.f90 File Reference 34
4.6.1 Function/Subroutine Documentation 35
4.6.1.1 ddot_par 35
4.7 dnrm2_par.f90 File Reference 35
4.7.1 Function/Subroutine Documentation 35
4.7.1.1 dnrm2_par 36
4.8 resid_par.f90 File Reference 36
4.8.1 Function/Subroutine Documentation 36
4.8.1.1 resid_par 36

Chapter 1

Data Type Index

1.1 Data Types List

Here are the data types with brief descriptions:

class_CSRMAT::assignment(=) 5
class_CSRMAT
Data type to store a CSR matrix 6
class_FSAIP_Prec
Data type to store a preconditioner as a pair of linked lists of CSR matrices 9
class_FSAIPACK
Module collecting all the FSAIPACK methods to compute sparse approximate inverses in factored form 16
class_Pattern
Data type to store the pattern of a CSR matrix 21
class_PCG_FSAIPACK
Defines the data structures needed by the Preconditioned Conjugate Gradient and implements the PCG solver 23
class_FSAIP_Prec::CSR_node
Typical node of the linked list of CSR matrices 26
class_CSRMAT::CSRMAT
Matrix stored in CSR format 27
class_FSAIP_Prec::FSAIP_Prec
Data type for the preconditioner 28
class_Pattern::Pattern
Pattern (p. 30) of a matrix in CSR format 30
class_PCG_FSAIPACK::PCG_FSAIPACK
PCG_FSAIPACK (p. 31) 31

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

A_PCG_FSAIPACK.f90 33
A_precond.f90 33
axbnsy.f90 33
B_CSRMAT.f90 34
B_FSAIPACK.f90 34
ddot_par.f90 34
dnrm2_par.f90 35
resid_par.f90 36

Chapter 3

Data Type Documentation

3.1 class_CSRMAT::assignment(=) Interface Reference

Public Member Functions

- subroutine **copy_CSRMAT** (mat_out, mat_in)

  Copy a CSR matrix data structure.

3.1.1 Member Function/Subroutine Documentation

3.1.1.1 subroutine class_CSRMAT::assignment(=)::copy_CSRMAT (type(CSRMAT), intent(inout) *mat_out,* type(CSRMAT), intent(in) *mat_in*)

Copy a CSR matrix data structure.

Author

Carlo Janna

Parameters

in	*mat_in*	**CSRMAT** (p. 27) variable
	inout]	mat_out **CSRMAT** (p. 27) variable

Version

1.0

Date

January 2013

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

- **B_CSRMAT.f90**

3.2 class_CSRMAT Module Reference

Data type to store a CSR matrix.

Data Types

- interface **assignment(=)**
- type **CSRMAT**

  matrix stored in CSR format

Public Member Functions

- integer function, public **new_CSRCOEF** (CSRMAT_inout)

  Allocates the CSR matrix coefficients.
- integer function, public **dlt_CSRCOEF** (CSRMAT_inout)

  Deallocates the CSR matrix coefficients.
- integer function, public **new_CSRMAT** (nrows, nterm, CSRMAT_inout)

  Allocates the CSR matrix data structure.
- integer function, public **dlt_CSRMAT** (CSRMAT_inout)

  Deallocates the CSR matrix data structure.
- subroutine, public **copy_CSRMAT** (mat_out, mat_in)

  Copy a CSR matrix data structure.
- subroutine, public **errchk_CSRMAT** (ounit, sub, ierr)

  Error handling routine.

3.2.1 Detailed Description

Data type to store a CSR matrix.

This module is used to define the Compact Sparse Row (CSR) data type for matrices storage (see Y. Saad -- Iterative Methods for Sparse Linear Systems)

Author

Carlo Janna

Version

1.0

Date

January 2013

License:

This program is intended for internal research only and can not be distributed elsewhere without authors' consent.

3.2.2 Member Function/Subroutine Documentation

3.2.2.1 subroutine, public class_CSRMAT::copy_CSRMAT (type(CSRMAT), intent(inout) *mat_out,* type(CSRMAT), intent(in) *mat_in*)

Copy a CSR matrix data structure.

Author

Carlo Janna

Parameters

in	*mat_in*	**CSRMAT** (p. 27) variable
	inout]	mat_out **CSRMAT** (p. 27) variable

Version

1.0

Date

January 2013

3.2.2.2 integer function, public class_CSRMAT::dlt_CSRCOEF (type(CSRMAT), intent(inout) *CSRMAT_inout*)

Deallocates the CSR matrix coefficients.

Author

Carlo Janna

Version

1.0

Date

January 2013

3.2.2.3 integer function, public class_CSRMAT::dlt_CSRMAT (type(CSRMAT), intent(inout) *CSRMAT_inout*)

Deallocates the CSR matrix data structure.

Author

Carlo Janna

Version

1.0

Date

January 2013

3.2.2.4 subroutine, public class_CSRMAT::errchk_CSRMAT (integer, intent(in) *ounit,* character∗(∗), intent(in) *sub,* integer, intent(in) *ierr*)

Error handling routine.

Author

Carlo Janna

Version

1.0

Date

January 2013

3.2.2.5 integer function, public class_CSRMAT::new_CSRCOEF (type(CSRMAT), intent(inout) *CSRMAT_inout*)

Allocates the CSR matrix coefficients.

Author

Carlo Janna

Version

1.0

Date

January 2013

3.2.2.6 integer function, public class_CSRMAT::new_CSRMAT (integer, intent(in) *nrows,* integer, intent(in) *nterm,* type(CSRMAT), intent(inout) *CSRMAT_inout*)

Allocates the CSR matrix data structure.

Author

Carlo Janna

Parameters

in	*nrows*	# of rows
in	*nterm*	# of non zeroes
	inout]	mat_inout **CSRMAT** (p. 27) variable
out	*ierr*	error code
	==	0, successful allocation
	/=	0, allocation error

Version

1.0

Date

January 2013

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

- **B_CSRMAT.f90**

3.3 class_FSAIP_Prec Module Reference

Data type to store a preconditioner as a pair of linked lists of CSR matrices.

Data Types

- type **CSR_node**

  Typical node of the linked list of CSR matrices.
- type **FSAIP_Prec**

  Data type for the preconditioner.

Public Member Functions

- integer function, public **append_LEFT** (CSR_in, PREC_list)

  Appends a CSR matrix at the end of the LEFT list.

- integer function, public **append_RIGHT** (CSR_in, PREC_list)

  Appends a CSR matrix at the beginning of the RIGHT list.
- integer function, public **delete_LEFT** (Content_Flag, PREC_list)

  Deletes the LEFT list.
- integer function, public **delete_RIGHT** (Content_Flag, PREC_list)

  Deletes the RIGHT list.
- integer function, public **dlt_FSAIP_Prec** (Content_Flag, PREC_list)

  Deletes the FSAIPACK preconditioner.
- subroutine, public **wr_FSAIP_Prec** (LEFT_flag, RIGHT_flag, PREC)

  Prints the FSAIPACK preconditioner in a sequence of files.
- subroutine, public **apply_FSAIP_Prec** (nrows, firstrow, PREC, vec, WR1, WR2, pvec)

  Applies the FSAIPACK preconditioner to a vector.
- subroutine, public **getdata_FSAIP_Prec** (PREC, nrows, nnz_L, nnz_R)

  Gets data from the preconditioner.

3.3.1 Detailed Description

Data type to store a preconditioner as a pair of linked lists of CSR matrices.

This module is used to create, store, apply and destroy the **FSAIP_Prec** (p. 28) data type. The preconditioner is stored as a pair of linked lists of CSR matrices. The first list, called LEFT, stores the left preconditioners [Fn]....[F2][F1], the second list, called RIGHT, stores the right preconditioners [F1^T][F2^T]....[Fn^T]. The preconditioned matrix is then: [Fn]....[F2][F1] [A] [F1^T]....[F2^T][F1^T], and the preconditioner is applied to a vector as:

pvec = [F1^T][F2^T]....[Fn^T] [Fn]....[F2][F1] vec.

Note: It is possible to use this data structure also with other preconditioners than those computed in the FSAIPACK. In such cases it is not needed that the CSR matrices in the RIGHT part are the Transpose of those in the LEFT and even their number may differ.

Author

Carlo Janna

Version

1.0

Date

February 2013

License:

This program is intended for internal research only and can not be distributed elsewhere without authors' consent.

3.3.2 Member Function/Subroutine Documentation

3.3.2.1 integer function, public class_FSAIP_Prec::append_LEFT (type(CSRMAT), intent(in), target *CSR_in,* type(FSAIP_Prec), intent(inout) *PREC_list*)

Appends a CSR matrix at the end of the LEFT list.

Author

Carlo Janna

Parameters

in	*CSR_in*	matrix to be appended
	inout]	PREC_list list of left preconditioners
out	*ierr*	error code
	==	0, successful allocation
	/=	0, allocation error

Version

1.0

Date

February 2013

3.3.2.2 integer function, public class_FSAIP_Prec::append_RIGHT (type(CSRMAT), intent(in), target *CSR_in,* type(FSAIP_Prec), intent(inout) *PREC_list*)

Appends a CSR matrix at the beginning of the RIGHT list.

Author

Carlo Janna

Parameters

in	*CSR_in*	matrix to be appended
	inout]	PREC_list list of right preconditioners
out	*ierr*	error code
	==	0, successful allocation
	/=	0, allocation error

Version

1.0

Date

February 2013

3.3.2.3 subroutine, public class_FSAIP_Prec::apply_FSAIP_Prec (integer, intent(in) *nrows,* integer, intent(in) *firstrow,* type(FSAIP_Prec), intent(in) *PREC,* real(kind=double), dimension(:), intent(in) *vec,* real(kind=double), dimension(:), intent(out), target *WR1,* real(kind=double), dimension(:), intent(out), target *WR2,* real(kind=double), dimension(:), intent(out), target *pvec*)

Applies the FSAIPACK preconditioner to a vector.

Author

Carlo Janna

Parameters

in	*nrows*	# of rows of the current stripe
in	*firstrow*	first row of the current stripe
in	*PREC*	FSAIPACK preconditioner
in	*vec*	vector to which the preconditioner is applied (local size)
out	*pvec*	preconditioned vector (local size)
out	*WR*	scratch array (global size)

Version

1.0

Date

February 2013

Here is the call graph for this function:

3.3.2.4 integer function, public class_FSAIP_Prec::delete_LEFT (logical, intent(in) *Content_Flag,* type(FSAIP_Prec), intent(inout) *PREC_list*)

Deletes the LEFT list.

Author

Carlo Janna

Parameters

in	*Content_-Flag\n*	
	.True.,delete	the list and its content
	.-False.,delete	the list only
	inout]	PREC_list list of left preconditioners to be deleted
out	*ierr*	error code
	==	0, successful allocation
	/=	0, allocation error

Version

1.0

Date

February 2013

3.3.2.5 integer function, public class_FSAIP_Prec::delete_RIGHT (logical, intent(in) *Content_Flag,* type(FSAIP_Prec), intent(inout) *PREC_list*)

Deletes the RIGHT list.

Author

Carlo Janna

Parameters

in	*Content_-Flag\n*	
	.True.,delete	the list and its content
	.-False.,delete	the list only
	inout]	PREC_list list of left preconditioners to be deleted
out	*ierr*	error code
	==	0, successful allocation
	/=	0, allocation error

Version

1.0

Date

February 2013

3.3.2.6 integer function, public class_FSAIP_Prec::dlt_FSAIP_Prec (logical, intent(in) *Content_Flag,* type(FSAIP_Prec), intent(inout) *PREC_list*)

Deletes the FSAIPACK preconditioner.

Author

Carlo Janna

Parameters

in	*Content_-Flag\n*	
	.True.,delete	the list and its content
	.-False.,delete	the list only
	inout]	PREC_list FSAIPACK preconditioner
out	*ierr*	error code
	==	0, successful allocation
	/=	0, allocation error

Version

1.0

Date

February 2013

3.3.2.7 subroutine, public class_FSAIP_Prec::getdata_FSAIP_Prec (type(FSAIP_Prec), intent(in) *PREC,* integer, intent(out) *nrows,* integer, intent(out) *nnz_L,* integer, intent(out) *nnz_R*)

Gets data from the preconditioner.

Author

Carlo Janna

Parameters

in	*PREC*	preconditioner to get data from
out	*nrows*	> 0 ---> # of rows of the preconditioner
	=	0 ---> there is no preconditioner stored
	<	0 ---> inconsistency between the parts of the preconditioner
out	*nnz_L*	# of non-zeroes of the left preconditioner
out	*nnz_R*	# of non-zeroes of the right preconditioner

Version

1.0

Date

February 2013

3.3.2.8 subroutine, public class_FSAIP_Prec::wr_FSAIP_Prec (logical, intent(in) *LEFT_flag,* logical, intent(in) *RIGHT_flag,* type(FSAIP_Prec), intent(in) *PREC*)

Prints the FSAIPACK preconditioner in a sequence of files.

Author

Carlo Janna

Parameters

in	*PREC*	preconditioner to print
in	*LEFT_flag*	flag to print or not the left part
in	*RIGHT_flag*	flag to print or not the right part

Version

1.0

Date

February 2013

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

- **A_precond.f90**

3.4 class_FSAIPACK Module Reference

Module collecting all the FSAIPACK methods to compute sparse approximate inverses in factored form.

Public Member Functions

- integer function **MK_patt** (tau_in, mu_min_in, kpow_in, mu_max_in, parDTSTR, A, PATTERN_OUT, PATTERN_IN)

  Pattern construction for static FSAI computation.
- integer function **CPT_static_FSAI** (parDTSTR, A, PATTERN_IN, FSAI)

  Computes the FSAI coefficients for a statically given input pattern.
- integer function **CPT_adapt_FSAI** (n_step_in, step_size_in, tau_in, eps_in, parDTSTR, A, FSAI)

  Computes a FSAI preconditioner by an adaptive procedure optionally starting from an input factor.
- integer function **CPT_projection_FSAI** (n_iter_in, nt_ret_in, tau_in, eps_in, parDTSTR, A, FSAI, F, FT)

  Computes the FSAI preconditioner through a projective strategy. In the present version a dropped Steepest Descent is used.
- integer function **FILTER_FSAI** (nnzr_max_in, tau_in, parDTSTR, A, FSAI)

  Filters a given FSAI preconditioner neglecting smallest elements.
- integer function **CPT_preconditioned_Matix** (nnzr_max_in, tau_in, parDTSTR, A, G, GT, B)

  Computes a dropped preconditioned matrix B = drop{[G][A][G]T}, where G is a previously computed FSAI preconditioner for A.
- integer function **TRANSPOSE_FSAI** (parDTSTR, FSAI, FSAI_T)

  Transposes a given FSAI preconditioner.
- subroutine **ERRCHK_FSAIPACK** (ounit, method_name, ierr)

  Error interpreter for the FSAIPACK methods.

3.4.1 Detailed Description

Module collecting all the FSAIPACK methods to compute sparse approximate inverses in factored form.

This module contains all the methods used to compute sparse approximate inverses in factored form. All the algorithms presented in [1] are available.

[1] 'FSAIPACK: a software package for high performance FSAI preconditioning', C. - Janna et al., ACM Transactions on Mathematical Software, submitted.

Author

Carlo Janna

Version

1.0

Date

April 2013

License:

This program is intended for internal research only and can not be distributed elsewhere without authors' consent.

3.4.2 Member Function/Subroutine Documentation

3.4.2.1 integer function class_FSAIPACK::CPT_adapt_FSAI (integer, intent(in) *n_step_in,* integer, intent(in) *step_size_in,* real(kind=double), intent(in) *tau_in,* real(kind=double), intent(in) *eps_in,* type(OMPDTSTR), intent(in) *parDTSTR,* type(CSRMAT), intent(in) *A,* type(CSRMAT), intent(inout) *FSAI*)

Computes a FSAI preconditioner by an adaptive procedure optionally starting from an input factor.

Computes a FSAI preconditioner by an adaptive procedure optionally starting from an input factor. If the FSAI variable is already allocated on entry, it is used as starting point for the adaptive procedure.

Author

Carlo Janna

Version

1.0

Date

February 2013

License:

This program is intended for internal research only and can not be distributed elsewhere without authors' consent.

3.4.2.2 integer function class_FSAIPACK::CPT_preconditioned_Matix (integer, intent(in) *nnzr_max_in,* real(kind=double), intent(in) *tau_in,* type(OMPDTSTR), intent(in) *parDTSTR,* type(CSRMAT), intent(in) *A,* type(CSRMAT), intent(in) *G,* type(CSRMAT), intent(in) *GT,* type(CSRMAT), intent(inout) *B*)

Computes a dropped preconditioned matrix B = drop{[G][A][G]T}, where G is a previously computed FSAI preconditioner for A.

Computes a dropped preconditioned matrix B = drop{[G][A][G]T}, where G is a previously computed FSAI preconditioner for A.

Author

Carlo Janna

Version

1.0

Date

February 2013

License:

This program is intended for internal research only and can not be distributed elsewhere without authors' consent.

3.4.2.3 integer function class_FSAIPACK::CPT_projection_FSAI (integer, intent(in) *n_iter_in,* integer, intent(in) *nt_ret_in,* real(kind=double), intent(in) *tau_in,* real(kind=double), intent(in) *eps_in,* type(OMPDTSTR), intent(in) *parDTSTR,* type(CSRMAT), intent(in) *A,* type(CSRMAT), intent(inout) *FSAI,* type(CSRMAT), intent(in), optional *F,* type(CSRMAT), intent(in), optional *FT*)

Computes the FSAI preconditioner through a projective strategy. In the present version a dropped Steepest Descent is used.

Computes the FSAI preconditioner through a projective strategy. In the present version a dropped Steepest Descent is used. It is possible to start from a tentative FSAI preconditione given as optional input as well as using an input FSAI to precondition the Steepest Descent. If the FSAI variable is already allocated on entry, it is used as starting point for the iterative procedure.

Note: the program will work correctly even if F and FT are not lower and upper triangular, they only need to represent a proper preconditioner for A.

Author

Carlo Janna

Version

1.0

Date

February 2013

License:

This program is intended for internal research only and can not be distributed elsewhere without authors' consent.

3.4.2.4 integer function class_FSAIPACK::CPT_static_FSAI (type(OMPDTSTR), intent(in) *parDTSTR,* type(CSRMAT), intent(in) *A,* type(Pattern), intent(in) *PATTERN_IN,* type(CSRMAT), intent(inout) *FSAI*)

Computes the FSAI coefficients for a statically given input pattern.

This function computes the coefficients of FSAI corresponding to previously computed pattern given as Input. The content of the variable used as Input pattern becomes part of the preconditioner variable. If the variable for the preconditioner contains some data on entry, it is deleted before computation.

Author

Carlo Janna

Version

1.0

Date

February 2013

License:

This program is intended for internal research only and can not be distributed elsewhere without authors' consent.

3.4.2.5 subroutine class_FSAIPACK::ERRCHK_FSAIPACK (integer, intent(in) *ounit,* character(len=*), intent(in) *method_name,* integer, intent(in) *ierr*)

Error interpreter for the FSAIPACK methods.

Interprets and prints to an output unit the error messages from the FSAIPACK methods.

Author

Carlo Janna

Version

1.0

Date

April 2013

License:

This program is intended for internal research only and can not be distributed elsewhere without authors' consent.

3.4.2.6 integer function class_FSAIPACK::FILTER_FSAI (integer, intent(in) *nnzr_max_in,* real(kind=double), intent(in) *tau_in,* type(OMPDTSTR), intent(in) *parDTSTR,* type(CSRMAT), intent(in) *A,* type(CSRMAT), intent(inout) *FSAI*)

Filters a given FSAI preconditioner neglecting smallest elements.

Filters a given FSAI preconditioner neglecting smallest elements with a dual drop strategy. The input preconditioner is replaced by a sparser one in the same variable.

Author

Carlo Janna

Version

1.0

Date

February 2013

License:

This program is intended for internal research only and can not be distributed elsewhere without authors' consent.

3.4.2.7 integer function class_FSAIPACK::MK_patt (real(kind=double), intent(in) *tau_in,* real(kind=double), intent(in) *mu_min_in,* integer, intent(in) *kpow_in,* real(kind=double), intent(in) *mu_max_in,* type(OMPDTSTR), intent(in) *parDTSTR,* type(CSRMAT), intent(in) *A,* type(Pattern), intent(inout) *PATTERN_OUT,* type(Pattern), intent(in), optional *PATTERN_IN*)

Pattern construction for static FSAI computation.

This function creates a pattern for static FSAI computation. First a prefiltration of the input matrix A is applied. If the density of {A} (A after prefiltration) is lower then mu_min, then the prefiltration tolerance is reduced. Then it computes kpow-th times power_PATT: {A}∗LOW({A}∗LOW({A}∗LOW({A}∗LOW({A})))) where LOW(∗) is a matrix function that returns the lower part of a matrix. This procedure terminates also if the density mu_max is reached. If the optional parameter PATTERN_IN is present, the function skips prefiltration and only the kpow computation is executed.

Author

Carlo Janna

Version

1.0

Date

January 2013

License:

This program is intended for internal research only and can not be distributed elsewhere without authors' consent.

3.4.2.8 integer function class_FSAIPACK::TRANSPOSE_FSAI (type(OMPDTSTR), intent(in) *parDTSTR,* type(CSRMAT), intent(in) *FSAI,* type(CSRMAT), intent(inout) *FSAI_T*)

Transposes a given FSAI preconditioner.

Transposes a given FSAI preconditioner.

Author

Carlo Janna

Version

1.0

Date

February 2013

License:

This program is intended for internal research only and can not be distributed elsewhere without authors' consent.

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

- **B_FSAIPACK.f90**

3.5 class_Pattern Module Reference

Data type to store the pattern of a CSR matrix.

Data Types

- type **Pattern**

  Pattern *(*p. *30) of a matrix in CSR format.*

Public Member Functions

- integer function, public **new_Pattern** (nrows, nterm, Pattern_inout)

  *Allocates the **Pattern** (p. 30) data structure.*
- integer function, public **dlt_Pattern** (Pattern_inout)

  *deallocates the **Pattern** (p. 30) data structure*

3.5.1 Detailed Description

Data type to store the pattern of a CSR matrix.

This module is used to create, store and destroy the non-zero pattern of a matrix stored in the Compact Sparse Row format.

Author

Carlo Janna

Version

1.0

Date

January 2013

License:

This program is intended for internal research only and can not be distributed elsewhere without authors' consent.

3.5.2 Member Function/Subroutine Documentation

3.5.2.1 integer function, public class_Pattern::dlt_Pattern (type(Pattern), intent(inout) *Pattern_inout*)

deallocates the **Pattern** (p. 30) data structure

Author

Carlo Janna

Version

1.0

Date

January 2013

3.5.2.2 integer function, public class_Pattern::new_Pattern (integer, intent(in) *nrows,* integer, intent(in) *nterm,* type(Pattern), intent(inout) *Pattern_inout*)

Allocates the **Pattern** (p. 30) data structure.

Author

Carlo Janna

Parameters

in	*nrows*	# of rows
in	*nterm*	# of non zeroes
	inout]	Pattern_inout **Pattern** (p. 30) variable
out	*ierr*	error code
	==	0, successful allocation
	/=	0, allocation error

Version

1.0

Date

January 2013

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

- **B_CSRMAT.f90**

3.6 class_PCG_FSAIPACK Module Reference

Defines the data structures needed by the Preconditioned Conjugate Gradient and implements the PCG solver.

Data Types

- type **PCG_FSAIPACK**

  PCG_FSAIPACK *(p. 31).*

Public Member Functions

- integer function, public **set_PCG_FSAIPACK** (iout, itmax, isol, tol_CG, PCG_var)

  *Allocates the **PCG_FSAIPACK** (p. 31) data structure.*

- integer function, public **solv_PCG_FSAIPACK** (parDTSTR, mat_A, PREC, PCG_var, rhs, sol)

  solve the system by PCG
- subroutine, public **errchk_PCG_FSAIPACK** (iunit, sub, ierr)

  Error handling subroutine.

3.6.1 Detailed Description

Defines the data structures needed by the Preconditioned Conjugate Gradient and implements the PCG solver.

This module defines the data structures needed by the Preconditioned Conjugate - Gradient solver (PCG) and contains all the methods to initialize, run and destroy PCG.

Author

Carlo Janna

Version

1.0

Date

February 2013

License:

This program is intended for internal research only and can not be distributed elsewhere without authors' consent.

3.6.2 Member Function/Subroutine Documentation

3.6.2.1 subroutine, public class_PCG_FSAIPACK::errchk_PCG_FSAIPACK (integer, intent(in) *iunit,* character(len=∗), intent(in) *sub,* integer, intent(in) *ierr*)

Error handling subroutine.

Author

Carlo Janna

Version

1.0

Date

January 2013

3.6.2.2 integer function, public class_PCG_FSAIPACK::set_PCG_FSAIPACK (integer, intent(in) *iout,* integer, intent(in) *itmax,* integer, intent(in) *isol,* real(kind=double), intent(in) *tol_CG,* type(**PCG_FSAIPACK**), intent(inout) *PCG_var*)

Allocates the **PCG_FSAIPACK** (p. 31) data structure.

Author

Carlo Janna

Parameters

in	*iout*	# output unit
in	*itmax*	max # of PCG iterations
in	*isol*	parameter to choose the starting solution
in	*tol_CG*	exit tolerance for PCG
	inout]	PCG_var **PCG_FSAIPACK** (p. 31) variable
out	*ierr*	error code
	==	0, successful allocation
	==	1, wrong input data

Version

1.0

Date

January 2013

3.6.2.3 integer function, public class_PCG_FSAIPACK::solv_PCG_FSAIPACK (type(OMPDTSTR), intent(in) *parDTSTR,* type(CSRMAT), intent(in) *mat_A,* type(FSAIP_Prec), intent(in) *PREC,* type(**PCG_FSAIPACK**), intent(inout) *PCG_var,* real(kind=double), dimension(mat_a%patt%nrows), intent(in) *rhs,* real(kind=double), dimension(mat_a%patt%nrows), intent(out) *sol*)

solve the system by PCG

Author

Carlo Janna

Parameters

in	*parDTSTR*	data type for the parallel handling of vectors and matrices
in	*mat_A*	input matrix in CSR format
in	*PREC*	preconditioner for matrix mat_A
	inout]	PCG_var **PCG_FSAIPACK** (p. 31) variable
in	*rhs*	right-hand side vector

	inout]	sol solution vector
out	*ierr*	error code
	==	0, successful solution
	==	1, inconsistency between mat_Apattnrows, PREC and par-DTSTR
	==	3, error allocating local arrays
	==	4, zero right-hand side
	==	5, convergence not achieved
	==	6, error deallocating local arrays

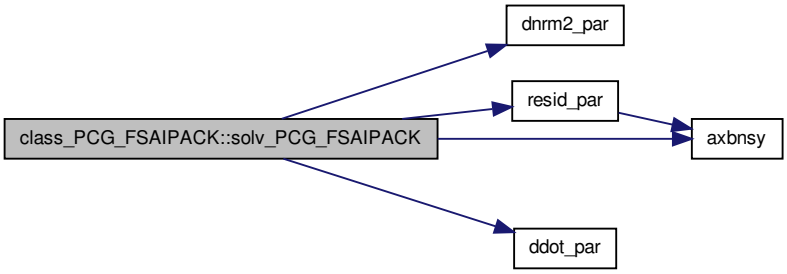
Version

1.0

Date

February 2013

Here is the call graph for this function:



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

- **A_PCG_FSAIPACK.f90**

3.7 class_FSAIP_Prec::CSR_node Type Reference

Typical node of the linked list of CSR matrices.

Collaboration diagram for class_FSAIP_Prec::CSR_node:

Public Attributes

- type(CSRMAT), pointer **pt_CSR**

  Pointer to the typical CSR matrix.
- type(**CSR_node**), pointer **next**

  Pointer to the next node of the list.

3.7.1 Detailed Description

Typical node of the linked list of CSR matrices.

3.7.2 Member Data Documentation

3.7.2.1 type(CSR_node), pointer class_FSAIP_Prec::CSR_node::next

Pointer to the next node of the list.

3.7.2.2 type(CSRMAT), pointer class_FSAIP_Prec::CSR_node::pt_CSR

Pointer to the typical CSR matrix.

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

- **A_precond.f90**

3.8 class_CSRMAT::CSRMAT Type Reference

matrix stored in CSR format

Public Attributes

- type(Pattern) **patt**

  matrix pattern

- real(kind=double), dimension(:), pointer **coef** => null()

  matrix coefficients

3.8.1 Detailed Description

matrix stored in CSR format

3.8.2 Member Data Documentation

3.8.2.1 real(kind=double), dimension(:), pointer class_CSRMAT::CSRMAT::coef => null()

matrix coefficients

3.8.2.2 type(Pattern) class_CSRMAT::CSRMAT::patt

matrix pattern

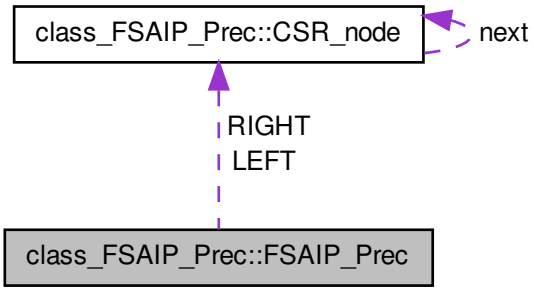
The documentation for this type was generated from the following file:

- **B_CSRMAT.f90**

3.9 class_FSAIP_Prec::FSAIP_Prec Type Reference

Data type for the preconditioner.

Collaboration diagram for class_FSAIP_Prec::FSAIP_Prec:



Public Attributes

- type(**CSR_node**), pointer **LEFT** => null()

  Pointer to the LEFT part of the preconditioner.
- type(**CSR_node**), pointer **RIGHT** => null()

  Pointer to the RIGHT part of the preconditioner.

3.9.1 Detailed Description

Data type for the preconditioner.

3.9.2 Member Data Documentation

3.9.2.1 type(CSR_node), pointer class_FSAIP_Prec::FSAIP_Prec::LEFT => null()

Pointer to the LEFT part of the preconditioner.

3.9.2.2 type(CSR_node), pointer class_FSAIP_Prec::FSAIP_Prec::RIGHT => null()

Pointer to the RIGHT part of the preconditioner.

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

- **A_precond.f90**

3.10 class_Pattern::Pattern Type Reference

Pattern (p. 30) of a matrix in CSR format.

Public Attributes

- integer **nrows** = 0

  number of rows
- integer **nterm** = 0

  number of non-zeroes
- integer, dimension(:), pointer **iat** = > null()

  pointers to the beginning of each row
- integer, dimension(:), pointer **ja** = > null()

  column indices of each row

3.10.1 Detailed Description

Pattern (p. 30) of a matrix in CSR format.

3.10.2 Member Data Documentation

3.10.2.1 integer, dimension(:), pointer class_Pattern::Pattern::iat = > null()

pointers to the beginning of each row

3.10.2.2 integer, dimension(:), pointer class_Pattern::Pattern::ja = > null()

column indices of each row

3.10.2.3 integer class_Pattern::Pattern::nrows = 0

number of rows

3.10.2.4 integer class_Pattern::Pattern::nterm = 0

number of non-zeroes

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

- **B_CSRMAT.f90**

3.11 class_PCG_FSAIPACK::PCG_FSAIPACK Type Reference

PCG_FSAIPACK (p. 31).

Public Attributes

- integer **iout**

  .gt. 0 ---> iout = output unit for the convergence profile .le. 0 ---> no convergence profile in output
- integer **itmax**

  max # of PCG iteration
- integer **isol**

  *.eq. 0 ---> use the Input solution vector as initial approximation .ne. 0 ---> compute the initial approximation as [PREC]^-1*b*
- integer **n_iter**

  # of iterations performed by PCG
- real(kind=double) **tol_CG**

  exit tolerance of PCG
- real(kind=double) **bnorm**

  rhs euclidean norm
- real(kind=double) **resini**

  initial relative residual
- real(kind=double) **resiter**

  iterative relative residual
- real(kind=double) **resreal**

  real relative residual

3.11.1 Detailed Description

PCG_FSAIPACK (p. 31).

3.11.2 Member Data Documentation

3.11.2.1 real(kind=double) class_PCG_FSAIPACK::PCG_FSAIPACK::bnorm

rhs euclidean norm

3.11.2.2 integer class_PCG_FSAIPACK::PCG_FSAIPACK::iout

.gt. 0 ---> iout = output unit for the convergence profile .le. 0 ---> no convergence profile in output

3.11.2.3 integer class_PCG_FSAIPACK::PCG_FSAIPACK::isol

.eq. 0 ---> use the Input solution vector as initial approximation .ne. 0 ---> compute the initial approximation as [PREC]^-1*b

3.11.2.4 integer class_PCG_FSAIPACK::PCG_FSAIPACK::itmax

max # of PCG iteration

3.11.2.5 integer class_PCG_FSAIPACK::PCG_FSAIPACK::n_iter

of iterations performed by PCG

3.11.2.6 real(kind=double) class_PCG_FSAIPACK::PCG_FSAIPACK::resini

initial relative residual

3.11.2.7 real(kind=double) class_PCG_FSAIPACK::PCG_FSAIPACK::resiter

iterative relative residual

3.11.2.8 real(kind=double) class_PCG_FSAIPACK::PCG_FSAIPACK::resreal

real relative residual

3.11.2.9 real(kind=double) class_PCG_FSAIPACK::PCG_FSAIPACK::tol_CG

exit tolerance of PCG

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

- **A_PCG_FSAIPACK.f90**

Chapter 4

File Documentation

4.1 A_PCG_FSAIPACK.f90 File Reference

Data Types

- module **class_PCG_FSAIPACK**

  Defines the data structures needed by the Preconditioned Conjugate Gradient and implements the PCG solver.
- type **class_PCG_FSAIPACK::PCG_FSAIPACK**

  PCG_FSAIPACK *(p. 31).*

4.2 A_precond.f90 File Reference

Data Types

- module **class_FSAIP_Prec**

  Data type to store a preconditioner as a pair of linked lists of CSR matrices.
- type **class_FSAIP_Prec::CSR_node**

  Typical node of the linked list of CSR matrices.
- type **class_FSAIP_Prec::FSAIP_Prec**

  Data type for the preconditioner.

4.3 axbnsy.f90 File Reference

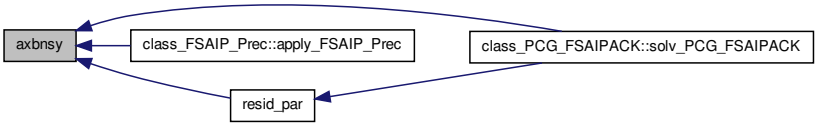
Functions/Subroutines

- subroutine **axbnsy** (n, ntot, nterm, iat, ja, coef_A, xvec, bvec)

4.3.1 Function/Subroutine Documentation

4.3.1.1 subroutine axbnsy (integer, intent(in) *n,* integer, intent(in) *ntot,* integer, intent(in) *nterm,* integer, dimension(n+1), intent(in) *iat,* integer, dimension(nterm), intent(in) *ja,* real(kind=double), dimension(nterm), intent(in) *coef_A,* real(kind=double), dimension(ntot), intent(in) *xvec,* real(kind=double), dimension(n), intent(out) *bvec*)

Here is the caller graph for this function:



4.4 B_CSRMAT.f90 File Reference

Data Types

- module **class_Pattern**

  Data type to store the pattern of a CSR matrix.
- type **class_Pattern::Pattern**

  Pattern (p. *30*) *of a matrix in CSR format.*
- module **class_CSRMAT**

  Data type to store a CSR matrix.
- type **class_CSRMAT::CSRMAT**

  matrix stored in CSR format
- interface **class_CSRMAT::assignment(=)**

4.5 B_FSAIPACK.f90 File Reference

Data Types

- module **class_FSAIPACK**

  Module collecting all the FSAIPACK methods to compute sparse approximate inverses in factored form.

4.6 ddot_par.f90 File Reference

Functions/Subroutines

- subroutine **ddot_par** (myid, nproc, nrows, v_loc, w_loc, ridv, fac, dotp)

4.6.1 Function/Subroutine Documentation

4.6.1.1 subroutine ddot_par (integer, intent(in) *myid,* integer, intent(in) *nproc,* integer, intent(in) *nrows,* real(kind=double), dimension(nrows), intent(in) *v_loc,* real(kind=double), dimension(nrows), intent(in) *w_loc,* real(kind=double), dimension(nproc), intent(inout) *ridv,* real(kind=double), intent(in) *fac,* real(kind=double), intent(out) *dotp*)

Here is the caller graph for this function:

4.7 dnrm2_par.f90 File Reference

Functions/Subroutines

- subroutine **dnrm2_par** (myid, nproc, nrows, v_loc, ridv, fac, norm)

4.7.1 Function/Subroutine Documentation

4.7.1.1 subroutine **dnrm2_par** (integer, intent(in) *myid,* integer, intent(in) *nproc,* integer, intent(in) *nrows,* real(kind=double), dimension(nrows), intent(in) *v_loc,* real(kind=double), dimension(nproc), intent(inout) *ridv,* real(kind=double), intent(in) *fac,* real(kind=double), intent(out) *norm*)

Here is the caller graph for this function:

4.8 resid_par.f90 File Reference

Functions/Subroutines

- subroutine **resid_par** (firstrow, nrows, nequ, nterm, iat, ja, coef_A, x_loc, vscr, vscr_loc, b_loc, r_loc)

4.8.1 Function/Subroutine Documentation

4.8.1.1 subroutine **resid_par** (integer, intent(in) *firstrow,* integer, intent(in) *nrows,* integer, intent(in) *nequ,* integer, intent(in) *nterm,* integer, dimension(nequ+1), intent(in) *iat,* integer, dimension(nterm), intent(in) *ja,* real(kind=double), dimension(nterm), intent(in) *coef_A,* real(kind=double), dimension(nrows), intent(in) *x_loc,* real(kind=double), dimension(nequ), intent(inout) *vscr,* real(kind=double), dimension(nrows), intent(inout) *vscr_loc,* real(kind=double), dimension(nrows), intent(in) *b_loc,* real(kind=double), dimension(nrows), intent(out) *r_loc*)

Here is the call graph for this function:

Here is the caller graph for this function: