## 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
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` .......................................................... 6
  Data type to store a CSR matrix ........................................
- `class_FSAIP_Prec` ..................................................... 9
  Data type to store a preconditioner as a pair of linked lists of CSR matrices ........................................
- `class_FSAIPACK` ....................................................... 16
  Module collecting all the FSAIPACK methods to compute sparse approximate inverses in factored form ...................
- `class_Pattern` .......................................................... 21
  Data type to store the pattern of a CSR matrix ......................
- `class_PCG_FSAIPACK` .................................................. 23
  Defines the data structures needed by the Preconditioned Conjugate Gradient and implements the PCG solver ............
- `class_FSAIP_Prec::CSR_node` ........................................ 26
  Typical node of the linked list of CSR matrices ........................
- `class_CSRMAT::CSRMAT` .............................................. 27
  Matrix stored in CSR format ............................................
- `class_FSAIP_Prec::FSAIP_Prec` ....................................... 28
  Data type for the preconditioner ......................................
- `class_Pattern::Pattern` .............................................. 30
  Pattern (p. 30) of a matrix in CSR format ............................
- `class_PCG_FSAIPACK::PCG_FSAIPACK` ................................ 31
  PCG_FSAIPACK (p. 31) ..................................................
Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

<table>
<thead>
<tr>
<th>File Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A_PCG_FSAIPACK.f90</td>
<td>33</td>
</tr>
<tr>
<td>A_precond.f90</td>
<td>33</td>
</tr>
<tr>
<td>axbnsy.f90</td>
<td>33</td>
</tr>
<tr>
<td>B_CSRMAT.f90</td>
<td>34</td>
</tr>
<tr>
<td>B_FSAIPACK.f90</td>
<td>34</td>
</tr>
<tr>
<td>ddot_par.f90</td>
<td>34</td>
</tr>
<tr>
<td>dnrm2_par.f90</td>
<td>35</td>
</tr>
<tr>
<td>resid_par.f90</td>
<td>36</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>in</th>
<th>mat_in</th>
<th>CSRMAT (p. 27) variable</th>
</tr>
</thead>
</table>

| inout | mat_out | CSRMAT (p. 27) variable |

Version

1.0

Date

January 2013

The documentation for this interface was generated from the following file:
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
3.2 class_CSRMAT Module Reference

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

<table>
<thead>
<tr>
<th>in</th>
<th>mat_in</th>
<th>CSRMB (p.27) variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>inout</td>
<td>mat_out</td>
<td>CSRMB (p.27) variable</td>
</tr>
</tbody>
</table>

Version
1.0

Date
January 2013

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nrows</td>
<td># of rows</td>
</tr>
<tr>
<td>nterm</td>
<td># of non zeroes</td>
</tr>
<tr>
<td>mat_inout</td>
<td>CSRMAT (p. 27) variable</td>
</tr>
<tr>
<td>ierr</td>
<td>error code</td>
</tr>
</tbody>
</table>

ierr == 0, successful allocation
ierr /= 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.
Data Type Documentation

- integer function, public `append_RIGHT` (CSR_in, PREC_list)
  Append 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 \([F1^\top][F2^\top][F1^\top]....[F1^\top][F2^\top][F1^\top]\). The preconditioned matrix is then: \([Fn]....[F2][F1][A][F1^\top][F2^\top][F1^\top][F2^\top][F1^\top]\), and the preconditioner is applied to a vector as:

\[ pvec = [F1^\top][F2^\top]....[F1^\top][F2][F1] vec. \]

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 class_FSAIP_Prec Module Reference

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

<table>
<thead>
<tr>
<th>in</th>
<th>CSR_in</th>
<th>matrix to be appended</th>
</tr>
</thead>
<tbody>
<tr>
<td>inout</td>
<td>PREC_list list of left preconditioners</td>
<td></td>
</tr>
<tr>
<td>out</td>
<td>ierr</td>
<td>error code</td>
</tr>
<tr>
<td></td>
<td>==</td>
<td>0, successful allocation</td>
</tr>
<tr>
<td></td>
<td>/=</td>
<td>0, allocation error</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>in</th>
<th>CSR_in</th>
<th>matrix to be appended</th>
</tr>
</thead>
<tbody>
<tr>
<td>inout</td>
<td>PREC_list list of right preconditioners</td>
<td></td>
</tr>
<tr>
<td>out</td>
<td>ierr</td>
<td>error code</td>
</tr>
<tr>
<td></td>
<td>==</td>
<td>0, successful allocation</td>
</tr>
<tr>
<td></td>
<td>/=</td>
<td>0, allocation error</td>
</tr>
</tbody>
</table>

Version
1.0
Applies the FSAIPACK preconditioner to a vector.

Author

Carlo Janna

Parameters

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>nrows</td>
<td># of rows of the current stripe</td>
</tr>
<tr>
<td>in</td>
<td>firstrow</td>
<td>first row of the current stripe</td>
</tr>
<tr>
<td>in</td>
<td>PREC</td>
<td>FSAIPACK preconditioner</td>
</tr>
<tr>
<td>in</td>
<td>vec</td>
<td>vector to which the preconditioner is applied (local size)</td>
</tr>
<tr>
<td>out</td>
<td>pvec</td>
<td>preconditioned vector (local size)</td>
</tr>
<tr>
<td>out</td>
<td>WR</td>
<td>scratch array (global size)</td>
</tr>
</tbody>
</table>

Version

1.0

Date

February 2013

Here is the call graph for this function:
3.3 class\_FSAIP\_Prec Module Reference

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

<table>
<thead>
<tr>
<th>in</th>
<th>Content_Flag\n</th>
<th>.True._delete</th>
<th>the list and its content</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>.</td>
<td>False_delete</td>
<td>the list only</td>
</tr>
<tr>
<td>inout)</td>
<td>PREC_list</td>
<td>list of left preconditioners to be deleted</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>out</th>
<th>ierr</th>
<th>error code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>== 0</td>
<td>successful allocation</td>
</tr>
<tr>
<td></td>
<td>/= 0</td>
<td>allocation error</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>in</th>
<th>Content_Flag\n</th>
<th>.True._delete</th>
<th>the list and its content</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>.</td>
<td>False_delete</td>
<td>the list only</td>
</tr>
<tr>
<td>inout)</td>
<td>PREC_list</td>
<td>list of left preconditioners to be deleted</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>out</th>
<th>ierr</th>
<th>error code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>== 0</td>
<td>successful allocation</td>
</tr>
<tr>
<td></td>
<td>/= 0</td>
<td>allocation error</td>
</tr>
</tbody>
</table>

Generated on Wed Jul 3 2013 11:13:09 for FSAIPACK Documentation by Doxygen
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

<table>
<thead>
<tr>
<th>in</th>
<th>Content_Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>.True.,_delete</td>
<td>the list and its content</td>
</tr>
<tr>
<td>False.,delete</td>
<td>the list only</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>inout</th>
<th>PREC_list FSAIPACK preconditioner</th>
</tr>
</thead>
<tbody>
<tr>
<td>ierr</td>
<td>error code</td>
</tr>
<tr>
<td>== 0</td>
<td>0, successful allocation</td>
</tr>
<tr>
<td>/= 0</td>
<td>0, allocation error</td>
</tr>
</tbody>
</table>

3.3.2.7  subroutine, public class FSAIP_Prec::getdata_FSAIP_Prec ( type(FSAIP_Prec), intent(in) PREC, integer, intent(out) nnrows, integer, intent(out) nnz_L, integer, intent(out) nnz_R )

Gets data from the preconditioner.

Author
Carlo Janna
3.3 class_FSAIP_Prec Module Reference

Parameters

<table>
<thead>
<tr>
<th>in</th>
<th>PREC</th>
<th>preconditioner to get data from</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>nrows</td>
<td>&gt; 0 --- &gt; # of rows of the preconditioner</td>
</tr>
<tr>
<td></td>
<td></td>
<td>= 0 --- &gt; there is no preconditioner stored</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt; 0 --- &gt; inconsistency between the parts of the preconditioner</td>
</tr>
<tr>
<td>out</td>
<td>nnz_L</td>
<td># of non-zeroes of the left preconditioner</td>
</tr>
<tr>
<td>out</td>
<td>nnz_R</td>
<td># of non-zeroes of the right preconditioner</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>in</th>
<th>PREC</th>
<th>preconditioner to print</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>LEFT_flag</td>
<td>flag to print or not the left part</td>
</tr>
<tr>
<td>in</td>
<td>RIGHT_flag</td>
<td>flag to print or not the right part</td>
</tr>
</tbody>
</table>

Version

1.0

Date

February 2013

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

- A_precond.f90

Generated on Wed Jul 3 2013 11:13:09 for FSAIPACK Documentation by Doxygen
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 $B = \text{drop}(G)[A][G]^T$, 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.


Author

Carlo Janna
3.4 class_FSAIPACK Module Reference

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 $B = \text{drop}([G][A][G]^T)$, 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) n_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 preconditioner 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 else-
where without authors' consent.

3.4.2.5 subroutine class_FSAIPACK::ERRCHK_FSAIPACK ( integer, intent(in) ouunit,
        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 else-
where 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_PATTERN: \{A\}+LOW( \{A\}+LOW( \{A\}+LOW( \{A\}+LOW(\{A\}))) where LOW(\cdot) 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
3.5 class_Pattern Module Reference

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.

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)

*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

<table>
<thead>
<tr>
<th>in</th>
<th>nrows</th>
<th># of rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>nterm</td>
<td># of non zeroes</td>
</tr>
<tr>
<td>inout</td>
<td>Pattern_inout Pattern (p. 30) variable</td>
<td></td>
</tr>
<tr>
<td>out</td>
<td>ierr</td>
<td>error code</td>
</tr>
<tr>
<td></td>
<td>== 0, successful allocation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>/= 0, allocation error</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>in</th>
<th>iout</th>
<th># output unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>itmax</td>
<td>max # of PCG iterations</td>
</tr>
<tr>
<td>in</td>
<td>isol</td>
<td>parameter to choose the starting solution</td>
</tr>
<tr>
<td>in</td>
<td>tol_CG</td>
<td>exit tolerance for PCG</td>
</tr>
<tr>
<td>inout</td>
<td>PCG_var</td>
<td>PCG_FSAIPACK (p. 31) variable</td>
</tr>
<tr>
<td>out</td>
<td>ierr</td>
<td>error code</td>
</tr>
<tr>
<td></td>
<td>== 0, successful allocation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>== 1, wrong input data</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>in</th>
<th>parDTSTR</th>
<th>data type for the parallel handling of vectors and matrices</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>mat_A</td>
<td>input matrix in CSR format</td>
</tr>
<tr>
<td>in</td>
<td>PREC</td>
<td>preconditioner for matrix mat_A</td>
</tr>
<tr>
<td>inout</td>
<td>PCG_var</td>
<td>PCG_FSAIPACK (p. 31) variable</td>
</tr>
<tr>
<td>in</td>
<td>rhs</td>
<td>right-hand side vector</td>
</tr>
</tbody>
</table>
### Data Type Documentation

<table>
<thead>
<tr>
<th>inout</th>
<th>sol solution vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>ierr error code</td>
</tr>
<tr>
<td></td>
<td>== 0, successful solution</td>
</tr>
<tr>
<td></td>
<td>== 1, inconsistency between mat_Apattnrows, PREC and par-DTSTR</td>
</tr>
<tr>
<td></td>
<td>== 3, error allocating local arrays</td>
</tr>
<tr>
<td></td>
<td>== 4, zero right-hand side</td>
</tr>
<tr>
<td></td>
<td>== 5, convergence not achieved</td>
</tr>
<tr>
<td></td>
<td>== 6, error deallocating local arrays</td>
</tr>
</tbody>
</table>

**Version**

1.0

**Date**

February 2013

Here is the call graph for this function:

```
class_PCG_FSAIPACK::solv_PCG_FSAIPACK
    dnrm2_par
    resid_par
    axbnsy
    ddot_par
```

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:

```
class_FSAIP_Prec::CSR_node  next
```

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) \texttt{patt}
  
  \textit{matrix pattern}

- real(kind=double), dimension(:,), pointer \texttt{coef} = \texttt{null()}
  
  \textit{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 \texttt{class\_CSRMAT::CSRMAT::coef} = \texttt{null()}

matrix coefficients

3.8.2.2 type(Pattern) \texttt{class\_CSRMAT::CSRMAT::patt}

matrix pattern

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

- \texttt{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:

![Collaboration diagram](image)

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`
  
  \[.gt. 0 \implies iout = output unit for the convergence profile \implies 0 \implies no convergence profile in output\]

- **integer** `itmax`
  
  max # of PCG iteration

- **integer** `isol`
  
  \[.eq. 0 \implies use the input solution vector as initial approximation \implies 0 \implies compute the initial approximation as \[\text{PREC}^\wedge 1+b\]\]

- **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

\[.gt. 0 \implies iout = output unit for the convergence profile \implies 0 \implies no convergence profile in output\]
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 $[\text{PREC}]^{-1}\times b$

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_PC_G_FSAIPACK.f90 File Reference

Data Types

- module class_PC_G_FSAIPACK
  Defines the data structures needed by the Preconditioned Conjugate Gradient and implements the PCG solver.
- type class_PC_G_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, intent(in) ia, integer, dimension(n), intent(in) ja, integer, dimension(nterm), intent(in) coeff_A, real(kind=double), dimension(nterm), intent(in) coef_A, real(kind=double), dimension(ntot), intent(in) coef_A, 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::CSRamat
  - 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:

![Caller Graph](ddot_par.png)

### 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 dnrm2Par (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:

```
    dnrm2Par  class_PCG_FSAIPACK::solve_PCG_FSAIPACK
```

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, 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(nrows), intent(in) x_scr, real(kind=double), dimension(nrows), intent(inout) vscr_loc, real(kind=double), dimension(nrows), intent(inout) b_loc, real(kind=double), dimension(nrows), intent(out) r_loc)

Here is the call graph for this function:

```
    resid_par  axbnsy
```
Here is the caller graph for this function:

```
resid_par  =>  class_PCG_FSAIPACK::solv_PCG_FSAIPACK
```