PSBLAS 90
Parallel Sparse
Basic Linear Algebra
Subroutines
Version 1.0

Contents

Chapter 1

F90 Data Structures

In this chapter are illustrated data structures used for definition of routines interfaces. This include data structure for sparse matrix and communication informations. These data structures are used for calling PSBLAS routines in Fortran 90 language and will be used to next chapters containing these callings. Their definitions are included in the modules `TYPESP` and `TYPE_PSP`.

Communication structure

Every structure of this type is associated to a sparse matrix, it contains data about general matrix informations and elements to be exchanged among processes. This structure will be named in next chapters as *decomp_data*.
It is not necessary for the user to know the internal structure of *decomp_data*, it is set in fully-transparent mode by PSBLAS-TOOLS routines when inserting a new sparse matrix, however the definition of the descriptor is the following.

matrix_data includes general information about matrix and BLACS grid. More precisely:

- **matrix_data[DEC_TYPE_]** Identifies the decomposition type (global); the actual values are internally defined, so they should never be accessed directly.
- **matrix_data[CTXT_]** Communication context as returned by the BLACS (global).
- **matrix_data[M_]** Total number of equations (global).
- **matrix_data[N_]** Total number of variables (global).
- **matrix_data[N_ROW_]** Number of grid variables owned by the current process (local); equivalent to the number of local rows in the sparse coefficient matrix.
- **matrix_data[N_COL_]** Total number of grid variables read by the current process (local); equivalent to the number of local columns in the sparse coefficient matrix. They include the halo.

Specified as: a pointer to integer array of dimension 10.

halo_index A list of the halo and boundary elements for the current process to be exchanged with other processes; for each processes with which it is necessary to communicate:

1. Process identifier;
2. Number of points to be received;
3. Indices of points to be received;
4. Number of points to be sent;
5. Indices of points to be sent;

The list may contain an arbitrary number of groups; its end is marked by a -1.
Specified as: a pointer to an integer array of rank one.

ovrlap_index A list of the overlap elements for the current process, organized in groups like the previous vector:

1. Process identifier;
2. Number of points to be received;
3. Indices of points to be received;
4. Number of points to be sent;
5. Indices of points to be sent;

The list may contain an arbitrary number of groups; its end is marked by a -1.
Specified as: a pointer to an integer array of rank one.

ovrlap_index For all overlap points belonging to th ecurrent process:

1. Overlap point index;
2. Number of processes sharing that overlap points;

The list may contain an arbitrary number of groups; its end is marked by a -1.
Specified as: a pointer to an integer array of rank one.

loc_to_glob each element i of this array contains global identifier of the local variable i.
Specified as: a pointer to an integer array of rank one.

glob_to_loc if global variable i is read by current process then element i contains local index correponding to global variable i; else element i contains $-(np+p+1)$ where np is the total number of processes and p is the index of the process owning global index i.
Specified as: a pointer to an integer array of rank one.

FORTRAN90 interface for *decomp_data* structures is therefore defined as follows:

```
 TYPE DECOMP_DATA_TYPE
     INTEGER, POINTER :: MATRIX_DATA(:)
     INTEGER, POINTER :: HALO_INDEX(:)
     INTEGER, POINTER :: OVRLAP_ELEM(:)
     INTEGER, POINTER :: OVRLAP_INDEX(:)
     INTEGER, POINTER :: LOC_TO_GLOB(:)
     INTEGER, POINTER :: GLOB_TO_LOC (:)
  END TYPE DECOMP_DATA_TYPE
```

Sparse matrix structure

Contains all information about local portion of the sparse matrix and its storage mode. Many of this fields are set in fully-transparent mode by PSBLAS-TOOLS routines when inserting a new sparse matrix, user must set only fields which describe matrix storage mode (see § 3).
Fields contained in Sparse matrix structures are:

ASPK Contains values of the local distributed sparse matrix.
Specified as: a pointer to an array of rank one of type corresponding to matrix entries type .

IA1 Holds integer information on distributed sparse matrix. Actual information will depend on data format used.
Specified as: a pointer to an integer array of rank one.

IA2 Holds integer information on distributed sparse matrix. Actual information will depend on data format used.
Specified as: a pointer to an integer array of rank one.

INFOA On entry can hold auxiliary information on distributed sparse matrix. Actual information will depend on data format used.
Specified as: integer array of length 10.

FIDA Defines the format of the distributed sparse matrix.
Specified as: a string of length 5

DESCRA Describe the characteristic of the distributed sparse matrix.
Specified as: array of character of length 9.

PL Specifies the local row permutation of distributed sparse matrix. If PL(1) is equal to 0, then there isn't row permutation.
Specified as: pointer to integer array of dimension equal to number of local row (matrix_data[N_ROW_])

PR Specifies the local column permutation of distributed sparse matrix. If PR(1) is equal to 0, then there isn't columnm permutation.

Specified as: pointer to integer array of dimension equal to number of local row (matrix_data[N_COL_])

M Number of rows; if row indices are stored explicitly, as in Coordinate Storage, should be greater than or equal to the maximum row index actually present in the sparse matrix. Specified as: integer variable.

K Number of columns; if column indices are stored explicitly, as in Coordinate Storage or Compressed Sparse Rows, should be greater than or equal to the maximum column index actually present in the sparse matrix. Specified as: integer variable.

Values assumed by this fields are compatible with ref. 1 (see § 4).
FORTRAN90 interface for distributed sparse matrices containing double precision real entries is defined as follows:

```
TYPE D_SPMAT
     INTEGER            :: M, K
     CHARACTER(LEN=5)   :: FIDA
     CHARACTER(LEN=11)  :: DESCRA
     INTEGER            :: INFOA(10)
     REAL(KIND(1.D0)), POINTER :: ASPK(:)
     INTEGER, POINTER :: IA1(:), IA2(:)
     INTEGER, POINTER :: PL(:), PR(:)
END TYPE D_SPMAT
```

The following two cases are used in the data insertion routines:

FIDA="CSR" Compressed storage rows. In this case the following should hold:

1. `IA2(I)` contains the index of the first element of row `I`; the last element of the sparse matrix is thus stored at index $IA2(M+1)-1$. It should contain `M+1` entries in nondecreasing order (strictly increasing, if there are no empty rows).
2. `IA1(J)` contains the column index and `ASPK(J)` contains the corresponding coefficient value, for all $IA2(1) \leq J \leq IA2(M+1)-1$.

FIDA="COO" Coordinate storage. In this case the following should hold:

1. `INFOA(1)` contains the number of nonzero elements in the matrix;
2. For all $1 \leq J \leq INFOA(1)$, the coefficient, row index and column index are stored into `APSK(J)`, `IA1(J)` and `IA2(J)` respectively.

Chapter 2

F90 PSBLAS Library

Routines in this chapter provide a convenient F90 interface to the computational kernels of PSBLAS; they hide most of the details depending on the parallel programming environment and on sparse matrix representation. Their interfaces are defined in the module `F90PSBLAS`.

Conventions on routines name

The name of a subroutine follow these conventions:

1. First five characters are **F90_PS** (that stands for *Parallel Sparse*)
2. Last characters describes the function performed by subroutine.

   The interfaces are constructed so that the compiler substitute all interfaces with the effective routines accordingly with type of parameters.

F90_PSAXPBY—General Dense Matrix Sum

This subroutine is an interface to the computational kernel for dense matrix sum:

$$Y \leftarrow \alpha X + \beta Y$$

where:

X represents the global dense submatrix $X_{:,jx:jx+n-1}$

Y represents the global dense submatrix $Y_{:,jy:jy+n-1}$

Syntax

CALL F90_PSAXPBY (*alpha, x, beta, y, decomp_data*)

CALL F90_PSAXPBY (*alpha, x, beta, y, decomp_data, n, jx, jy*)

X, Y, α, β	**Subroutine**
Long Precision Real	F90_PSAXPBY
Long Precision Complex	F90_PSAXPBY

Table 2.1: Data types

On Entry

alpha the scalar α.
Scope: **global**
Type: **required**
Specified as: a number of the data type indicated in Table 2.1.

x the local portion of global dense matrix X.
Scope: **local**
Type: **required**
Specified as: a rank one or two array containing numbers of type specified in Table 2.1. The rank of x must be the same of y.

beta the scalar β.
Scope: **global**
Type: **required**
Specified as: a number of the data type indicated in Table 2.1.

y the local portion of the global dense matrix Y.
Scope: **local**
Type: **required**
Specified as: a rank one or two array with the POINTER attributecontaining numbers of the type indicated in Table 2.1. The rank of y must be the same of x.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

n number of columns in dense submatrices X and Y.
Scope: **global**
Type: **optional**; can only be present if x and y are of rank 2.
Default: `min(size(X,2),size(Y,2))`.
Specified as: an integer variable $n \geq 0$.

jx the column index of the global dense matrix X, identifying the first column of the submatrix X.
Scope: **global**
Type: **optional**; can only be present if x and y are of rank 2.
Default: $jx = 1$.
Specified as: an integer variable $jx \geq 1$.

jy the column index of the global dense matrix Y, identifying the first column of the submatrix Y.
Scope: **global**
Type: **optional**; can only be present if x and y are of rank 2.
Default: $jy = 1$.
Specified as: an integer variable $jy \geq 1$.

On Return

y the local portion of result submatrix Y.
Scope: **local**
Type: **required**
Specified as: a rank one or two array containing numbers of the type indicated in Table 2.1.

F90_PSDOT—Dot Product

This function computes dot product between two vectors X and Y.
If X and Y are double precision or complex vectors computes dot-product as:

$$dot \leftarrow X^T Y$$

Else if X and Y are double precision complex vectors then computes dot-product as:

$$dot \leftarrow X^H Y$$

where:

X represents the global subvector $X_{:,jx}$

Y represents the global subvector $Y_{:,jy}$

Syntax

F90_PSDOT (*x, y, decomp_data*)

F90_PSDOT (*x, y, decomp_data, jx, jy*)

dot, X, Y	**Function**
Long Precision Real	F90_PSDOT
Long Precision Complex	F90_PSDOT

Table 2.2: Data types

On Entry

x the local portion of global dense matrix X. This function computes the location of the first element of local subarray used, based on jx and the field *matrix_data* of *decomp_data* .
Scope: **local**
Type: **required**
Specified as: a pointer to array of rank one or two containing numbers of type specified in Table 2.2. The rank of x must be the same of y.

y the local portion of global dense matrix Y. This function computes the location of the first element of local subarray used, based on iy, jy and the field *matrix_data* of *decomp_data* .
Scope: **local**

Type: **required**
Specified as: a pointer to array of rank one or two containing numbers of type specified in Table 2.2. The rank of y must be the same of x.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

jx the column index of global dense matrix X, identifying the column of subvector X.
Scope: **global**
Type: **optional**; can only be present if x and y are of rank 2.
Default: $jx = 1$.

jy the column index of global dense matrix Y, identifying the column of subvector Y.
Scope: **global**
Type: **optional**; can only be present if x and y are of rank 2.
Default: $jy = 1$.
Specified as: an integer variable $jy \geq 1$.

On Return

Function value is the dot product of subvectors X and Y.
Scope: **global**
Specified as: a number of the data type indicated in Table 2.2.

F90_DOT—Generalized Dot Product

This subroutine computes a series of dot products among the columns of two dense matrices X and Y:

$$res(i) \leftarrow X(:,i)^T Y(:,i)$$

If the matrices are complex, then the usual convention applies, i.e. the conjugate transpose of X is used. If X and Y are of rank one, then res is a scalar, else it is a rank one array.

Syntax

F90_DOT (*res, x, y, decomp_data*)

res, X, Y	**Subroutine**
Long Precision Real	F90_DOT
Long Precision Complex	F90_DOT

Table 2.3: Data types

On Entry

x the local portion of global dense matrix X.
Scope: **local**
Type: **required**
Specified as: a pointer to array of rank one or two containing numbers of type specified in Table 2.3. The rank of x must be the same of y.

y the local portion of global dense matrix Y.
Scope: **local**
Type: **required**
Specified as: a pointer to array of rank one or two containing numbers of type specified in Table 2.3. The rank of y must be the same of x.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in Sec. 1.

On Return

res is the dot product of subvectors X and Y.

Scope: **global**

Specified as: a number or a rank-one array of the data type indicated in Table 2.2.

F90_PSAMAX—Infinity-Norm of Vector

This function computes the infinity-norm of a vector X.
If X is double precision real vector computes infinity norm as:

$$amax \leftarrow \max_i |x_i|$$

else if X is double precision complex vector then computes infinity-norm as:

$$amax \leftarrow \max_i (|re(x_i)| + |im(x_i)|)$$

where:

X represents the global subvector $X_{:,jx}$

Syntax

F90_PSAMAX (*x,decomp_data*)

F90_PSAMAX (*x,decomp_data, jx*)

amax	*X*	**Function**
Long Precision Real	Long Precision Real	F90_PSAMAX
Long Precision Real	Long Precision Complex	F90_PSZAMAX

Table 2.4: Data types

On Entry

x the local portion of global dense matrix X. This function computes the location of the first element of local subarray used, based on jx and the field *matrix_data* of *decomp_data* .
Scope: **local**
Type: **required**
Specified as: a rank one or two array containing numbers of type specified in Table 2.4.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

jx the column index of global dense matrix X, identifying the column of sub-vector X.
Scope: **global**
Type: **optional**; can only be present if x is of rank 2.
Default: $jx = 1$
Specified as: an integer variable $jx \geq 1$.

On Return

Function value is the infinity norm of subvector X.
Scope: **global**
Specified as: a number of the data type indicated in Table 2.4.

F90_AMAX—Generalized Infinity Norm

This subroutine computes a series of infinity norms on the columns of a dense matrix X:

$$res(i) \leftarrow \max_k |X(k,i)|$$

Syntax

F90_AMAX (*res, X, decomp_data*)

res	*X*	**Subroutine**
Long Precision Real	Long Precision Real	F90_AMAX
Long Precision Real	Long Precision Complex	F90_AMAX

Table 2.5: Data types

On Entry

x the local portion of global dense matrix X.
Scope: **local**
Type: **required**
Specified as: a rank one or two array containing numbers of type specified in Table 2.5.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in Sec. 1.

On Return

res is the infinity norm of the columns of X.
Scope: **global**
Specified as: a number or a rank-one array of the data type indicated in Table 2.4.

F90_PSASUM—1-Norm of Vector

This function computes the 1-norm of a vector X.
If X is double precision real vector computes 1-norm as:

$$asum \leftarrow \|x_i\|$$

else if X ic double precision complex vector then computes 1-norm as:

$$asum \leftarrow \|re(X)\|_1 + \|im(X)\|_1$$

where:

X represents the global subvector $X_{:,jx}$

Syntax

F90_PSASUM (*x, decomp_data*)

F90_PSASUM (*x, decomp_data, jx*)

dot, X, Y	**Function**
Long Precision Real	F90_PSASUM
Long Precision Complex	F90_PSASUM

Table 2.6: Data types

On Entry

x the local portion of global dense matrix X. This function computes the location of the first element of local subarray used, based on jx and the field *matrix_data* of *decomp_data* .
Scope: **local**
Type: **required**
Specified as: a rank one or two array containing numbers of type specified in Table 2.6.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

jx the column index of global dense matrix X, identifying the column of subvector X.
Scope: **global**
Type: **optional**; can only be present if x is of rank 2.
Default: $jx = 1$
Specified as: an integer variable $jx \geq 1$.

On Return

Function value is the 1-norm of subvector X.
Scope: **global**
Specified as: a number of the data type indicated in Table 2.6.

F90_PSNRM2—2-Norm of Vector

This function computes the 2-norm of a vector X.
If X is double precision real vector computes 2-norm as:

$$nrm2 \leftarrow \sqrt{X^T X}$$

else if X is double precision complex vector then computes 2-norm as:

$$nrm2 \leftarrow \sqrt{X^H X}$$

where:

X represents the global subvector $X_{:,jx}$

$nrm2$, X	**Function**
Long Precision Real	F90_PSNRM2
Long Precision Complex	F90_PSNRM2

Table 2.7: Data types

Syntax

F90_PSNRM2 (*x, decomp_data*)

F90_PSNRM2 (*x, decomp_data, jx*)

On Entry

x the local portion of global dense matrix X. This function computes the location of the first element of local subarray used, based on jx and the field *matrix_data* of *decomp_data* .
Scope: **local**
Type: **required**
Specified as: a rank one or two array containing numbers of type specified in Table 2.7.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

jx the column index of global dense matrix X, identifying the column of subvector X.
Scope: **global**
Type: **optional**; can only be present if x is of rank 2.
Default: $jx = 1$
Specified as: an integer variable $jx \geq 1$.

On Return

Function Value is the 2-norm of subvector X.
Scope: **global**
Type: **required**
Specified as: a number of the data type indicated in Table 2.7.

F90_PSNRMI—Infinity Norm of Sparse Matrix

This function computes the infinity-norm of a matrix A:

$$nrmi \leftarrow \|A\|_\infty$$

where:

A represents the global matrix A

$nrmi$, A	**Function**
Long Precision Real	F90_PSNRMI
Long Precision Complex	F90_PSNRMI

Table 2.8: Data types

Syntax

F90_PSNRMI (A, *decomp_data*)

On Entry

a the local portion of the global sparse matrix A.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

On Return

Function value is the infinity-norm of sparse submatrix A.
Scope: **global**
Specified as: a number of the data type indicated in Table 2.8.

F90_PSSPMM—Sparse Matrix by Dense Matrix Product

This subroutine computes the Sparse Matrix by Dense Matrix Product:

$$Y \leftarrow \alpha P_r A P_c X + \beta Y \tag{2.1}$$

$$Y \leftarrow \alpha P_r A^T P_c X + \beta Y \tag{2.2}$$

$$Y \leftarrow \alpha P_r A^H P_c X + \beta Y \tag{2.3}$$

where:

X is the global dense submatrix $X_{:,jx:jx+k-1}$

Y is the global dense submatrix $Y_{:,jy:jy+k-1}$

A is the global sparse submatrix A

P_r, P_c are the permutation matrices.

A, X, Y, α, β	**Subroutine**
Long Precision Real	F90_PSSPMM
Long Precision Complex	F90_PSSPMM

Table 2.9: Data types

Syntax

CALL F90_PSSPMM (*alpha, a, x, beta, y, decomp_data*)

CALL F90_PSSPMM (*alpha, a, x, beta, y,decomp_data, trans, k, jx, jy, work*)

On Entry

alpha the scalar α.
Scope: **global**
Type: **required**
Specified as: a number of the data type indicated in Table 2.9.

a the local portion of the sparse matrix A.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

x the local portion of global dense matrix X. This subroutine computes the location of the first element of local subarray used, based on jx and the field *matrix_data* of *decomp_data* .
Scope: **local**
Type: **required**
Specified as: a rank one or two array containing numbers of type specified in Table 2.9. The rank of x must be the same of y.

beta the scalar β.
Scope: **global**
Type: **required**
Specified as: a number of the data type indicated in Table 2.9.

y the local portion of global dense matrix Y. This subroutine computes the location of the first element of local subarray used, based on jy and the field *matrix_data* of *decomp_data* .
Scope: **local**
Type: **required**
Specified as: a rank one or two array containing numbers of type specified in Table 2.9. The rank of y must be the same of x.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

trans indicate what kind of operation to perform.

trans = N the operation is specified by equation 2.1

trans = T the operation is specified by equation 2.2

trans = C the operation is specified by equation 2.3

Scope: **global**
Type: **optional**
Default: $trans = N$
Specified as: a character variable.

k number of columns in dense submatrices X and Y.
Scope: **global**
Type: **optional**
Default: `min(size(x,2)-jx+1,size(y,2)-jy+1)`
Specified as: an integer variable $k \geq 1$.

jx the column index of global dense matrix X, identifying the column of subvector X.
Scope: **global**
Type: **optional**; can only be present if x is of rank 2.
Default: $iy = 1$
Specified as: an integer variable $jx \geq 1$.

jy the column index of global dense matrix Y, identifying the column of sub-vector Y.
Scope: **global**
Type: **optional**; can only be present if y is of rank 2.
Default: $jy = 1$
Specified as: an integer variable $jy \geq 1$.

work the work array.
Scope: **local**
Specified as: a rank one array of the same type of X and Y with the POINTER attribute.

On Return

y the local portion of result submatrix Y.
Scope: **local**
Type: **required**
Specified as: a pointer to array of rank one or two containing numbers of type specified in Table 2.9.

F90_PSSPSM—Triangular System Solve

This subroutine computes the Triangular System Solve:

$$
\begin{aligned}
Y &\leftarrow \alpha P_r T^{-1} P_c X + \beta Y \\
Y &\leftarrow \alpha D P_r T^{-1} P_c X + \beta Y \\
Y &\leftarrow \alpha P_r T^{-1} P_c D X + \beta Y \\
Y &\leftarrow \alpha P_r T^{-T} P_c X + \beta Y \\
Y &\leftarrow \alpha D P_r T^{-T} P_c X + \beta Y \\
Y &\leftarrow \alpha P_r T^{-T} P_c D X + \beta Y \\
Y &\leftarrow \alpha P_r T^{-H} P_c X + \beta Y \\
Y &\leftarrow \alpha D P_r T^{-H} P_c X + \beta Y \\
Y &\leftarrow \alpha P_r T^{-H} P_c D X + \beta Y
\end{aligned}
$$

where:

X is the global dense submatrix $X_{:,jx:jx+n-1}$

Y is the global dense submatrix $Y_{:,jy:jy+n-1}$

T is the global sparse block triangular submatrix T

D is the scaling diagonal matrix.

P_r, P_c are the permutation matrices.

Syntax

CALL F90_PSSPSM (*alpha, t, x, beta, y, decomp_data*)

CALL F90_PSSPSM
(*alpha, t, x, beta, y, decomp_data, trans, unit, choice, diag, n, jx, jy, work*)

T, X, Y, D, α, β	**Subroutine**
Long Precision Real	F90_PSSPSM
Long Precision Complex	F90_PSSPSM

Table 2.10: Data types

On Entry

alpha the scalar α.
Scope: **global**
Type: **required**
Specified as: a number of the data type indicated in Table 2.10.

t the global portion of the sparse matrix T.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

x the local portion of global dense matrix X. This subroutine computes the location of the first element of local subarray used, based on jx and the field *matrix_data* of *decomp_data* .
Scope: **local**
Type: **required**
Specified as: a rank one or two array containing numbers of type specified in Table 2.10. The rank of x must be the same of y.

beta the scalar β.
Scope: **global**
Type: **required**
Specified as: a number of the data type indicated in Table 2.10.

y the local portion of global dense matrix Y. This subroutine computes the location of the first element of local subarray used, based on jy and the field *matrix_data* of *decomp_data* .
Scope: **local**
Type: **required**
Specified as: a rank one or two array containing numbers of type specified in Table 2.10. The rank of y must be the same of x.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

trans specify with *unitd* the operation to perform.

trans = 'N' the operation is with no transposed matrix

trans = 'T' the operation is with transposed matrix.

trans = 'C' the operation is with conjugate transposed matrix.

Scope: **global**
Type: **optional**
Default: $trans = N$
Specified as: a character variable.

unitd specify with *trans* the operation to perform.

unitd = 'U' the operation is with no scaling

unitd = 'L' the operation is with left scaling

unitd = 'R' the operation is with right scaling.

Scope: **global**
Type: **optional**
Default: $unitd = U$
Specified as: a character variable.

choice specify whether a cleanup of the overlapped elements is required on exit.

choice = .false. no cleanup on exit

choice = .true. cleanup on exit.

Scope: **global**
Type: **optional**
Default: $choice = .true.$
Specified as: a logical variable.

diag the diagonal scaling matrix.
Scope: **local**
Type: **optional**
Default: $diag(1) = 1(noscaling)$
Specified as: a rank one array containing numbers of the type indicated in Table 2.10.

n number of columns in dense submatrices X and Y.
Scope: **global**
Type: **optional**
Default: `min(size(x,2)-jx+1,size(y,2)-jy+1)`
Specified as: an integer variable $n \geq 0$.

jx the column index of global dense matrix X, identifying the column of subvector X.
Scope: **global**
Type: **optional**; can only be present if x is of rank 2.
Default: $jx = 1$
Specified as: an integer variable $jx \geq 1$.

jy the column index of global dense matrix Y, identifying the column of subvector Y.
Scope: **global**
Type: **optional**; can only be present if y is of rank 2.
Default: $jy = 1$
Specified as: an integer variable $jy \geq 1$.

Scope: **global**
Specified as: a number of the data type indicated in Table 2.10.

work the work array.
Scope: **local**
Type: **optional**
Specified as: a rank one array of the same type of X with the POINTER attribute.

On Return

y the local portion of global dense matrix Y. This subroutine computes the location of the first element of local subarray used, based on jy and the field *matrix_data* of *decomp_data* .
Scope: **local**
Type: **required**
Specified as: a pointer to array of rank one or two containing numbers of type specified in Table 2.10.

F90_PSHALO—Halo Data Communication

These subroutines restore a consistent status for the halo elements, and (optionally) scale the result:

$$X \leftarrow \alpha X$$

where:

X is a global dense submatrix.

α, X	**Subroutine**
Integer	F90_PSHALO
Long Precision Real	F90_PSHALO
Long Precision Complex	F90_PSHALO

Table 2.11: Data types

Syntax

CALL F90_PSHALO (*x, decomp_data*)

CALL F90_PSHALO (*x, decomp_data, alpha, work*)

On Entry

x global dense matrix X.
Scope: **local**
Type: **required**
Specified as: a rank one or two array containing numbers of type specified in Table 2.11.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

alpha the scalar α.
Scope: **global**
Type: **optional**
Default: $alpha = 1$
Specified as: a number of the data type indicated in Table 2.11.

work the work array.
Scope: **local**
Type: **optional**
Specified as: a rank one array of the same type of X with the POINTER attribute.

On Return

x global dense result matrix X.
Scope: **local**
Type: **required**
Returned as: a rank one or two array containing numbers of type specified in Table 2.11.

F90_PSOVRL—Overlap Update

These subroutines restore a consistent status for the overlap elements:

$$X \leftarrow QX$$

where:

X is the global dense submatrix X

Q is the overlap operator; it is the composition of two operators P_a and P^T.

X	**Subroutine**
Long Precision Real	F90_PSOVRL
Long Precision Complex	F90_PSOVRL

Table 2.12: Data types

Syntax

CALL F90_PSOVRL (*x, decomp_data*)

CALL F90_PSOVRL (*x, decomp_data, CHOICE=choice, UPDATE_TYPE=update_type, WORK=work*)

On Entry

x global dense matrix X.
Scope: **local**
Type: **required**
Specified as: a rank one or two array containing numbers of type specified in Table 2.12.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1. item[choice] specify if exchange overlap elements.

choice = .true. exchange overlap elements, i.e. apply operator P^T;

choice = .false. don't exchange overlap elements

Scope: **global**
Type: **optional**
Default: *choice* = *.true*.
Specified as: a logical variable.

update_type = 1 normal update P_a;

update_type = 2 square root update $\sqrt{P_a}$;

Scope: **global**
Default: *update_type* = *.true*.
Scope: **global**
Specified as: a integer variable.

work the work array.
Scope: **local**
Type: **optional**
Specified as: a one dimensional array of the same type of X.

On Return

x global dense result matrix X.
Scope: **local**
Type: **required**
Specified as: a pointer to array of rank one or two containing numbers of type specified in Table 2.12.

Usage notes

1. If there is no overlap in the data distribution, no operations are performed;
2. The operator P^T performs the reduction sum of overlap elements; it is the inverse of a "stretch" operator P that replicates overlap elements, accounting for the physical replication of data;
3. The operator P_a performs a scaling on the overlap elements by the amount of replication; thus, when combined with the reduction operator, it implements the average of replicated elements over all of their instances.
4. The square root update option makes it possible to applythe following operator:
$$X \leftarrow \sqrt{P_a} P^T K^{-1} P \sqrt{P_a} X$$
In the case of a symmetric K, this preserves simmetry of the overall preconditioner, which would otherwise be destroyed.

Chapter 3

Tools Library

Routines in this chapter provide a convenient interface to the kernels for initialize and set communication data structures, sparse matrix and dense matrix; they hide most of the details depending on the parallel programming environment. Their interfaces are defined in the module `F90TOOLS` To use the TOOLS subroutines, the user must define a partition subroutine that specifies how global matrix rows are assigned over all processes.

PARTITION—User defined partition subroutine

Defines how global matrix rows are assigned over all processes. This routine must be implemented by user and must be passed as argument to all Tools library subroutine that require it.
It must satisfy the following specification:

```
interface
   subroutine partition(glob_index,nrow,np,pv,nv)
     integer  glob_index,np,nrow,nv
     integer  pv(*)
   end subroutine partition
end interface
```

On Entry

glob_index global matrix row index
Specified as: an integer variable $0 \leq glob\_index \leq No\_row$, where No_row si the total number of global matrix rows.

procs For return values see "On Return".
Specified as: integer array of length l with $l \geq N_p$.

nprocs see "On Return".

On Return

procs contains process identifier to which is assigned *glob_row* matrix row.
Specified as: integer array of length l with $l \geq N_p$.

nprocs No of process to which is assigned *glob_row* matrix row.
Specified as: integer $1 \leq procs \leq N_p$.

Usage Notes

1. Duplicate elements are not admitted on return array *procs*
2. If this subroutine returns for a row matrix r_m a number of processes $nprocs$ greater than 1, then r_m will be an overlap row between all processes that own it.
3. An example of partition subroutine which subdivide sparse matrix in block rows with an one row overlap of is:

Example 1

```
      SUBROUTINE PART_BLOCK(GLOB_INDX,PROCS,NPROCS)
      INTEGER      GLOB_INDX
      INTEGER      PROCS(*),NPROCS


C     ...Common area previously initialized ...
      COMMON /BLOCK_DATA/ DIM_BLOCK, NP
      INTEGER DIM_BLOCK, NP
      SAVE  /BLOCK_DATA/
C     ..DIM_BLOCK..is dimension (number of rows) of each matrix block
C     ..NP.........is number of processes involved in computation

      NPROCS=1
      PROCS(1) = (GLOB_INDX-1)/DIM_BLOCK
      IF ((MOD(GLOB_INDX, DIM_BLOCK).EQ.0).AND.
     + (PROCS(NPROCS).NE.(NP-1))) THEN
C       ....this is an overlap row....
          NPROCS = NPROCS+1
          PROCS(NPROCS) = PROCS(NPROCS-1)+1
      ENDIF
      RETURN

      BLOCK DATA BLOCK
      COMMON /BLOCK_DATA/ DIM_BLOCK, NP
      INTEGER DIM_BLOCK, NP
      SAVE  /BLOCK_DATA/
```

F90_PSDSCALL —Allocate Communication Descriptor

This subroutine allocates the communication desciptor for global sparse matrix A.

Syntax

CALL F90_PSDSCALL (*m,n,parts,icontxt, ierrv, decomp_data*)

On Entry

m number of rows of global sparse submatrix A to allocate.
Scope: **global**
Type: **required**
Specified as: an integer variable $0 \leq m \leq decomp\_data\%matrix\_data[M\_]$.

n number of columns of global sparse submatrix X to allocate.
Scope: **global**
Type: **required**
Specified as: an integer variable $n \geq 0$.

parts function that specify the partition of global matrix rows over all process. See ... for more informations.
Scope: **global**
Type: **required**
Specified as: a procedure.

icontxt BLACS communication context.
Scope: **global**
Type: **required**
Specified as: an integer variable.

ierrv see "On Return".

On Return

ierrv error vector used in subroutines F90_PSSPINS and F90_PSSPASB.
Scope: **local**
Type: **required**
Specified as: a one dimensional array of integer.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

F90_PSSPALL —Allocate Global Sparse Matrix

This subroutine allocates global sparse matrix A, the storage mode of matrix A is decided by user setting its FIDA and DESCRA fields.

A	**Subroutine**
Long Precision Real	F90_PSSPALL
Long Precision Complex	F90_PSSPALL

Table 3.1: Data types

Syntax

CALL F90_PSSPALL (*a, ierrv, decomp_data*)

CALL F90_PSSPALL (*a, ierrv, decomp_data, nnz*)

On Entry

a see "On Return".

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

ierrv see "On Return".

nnz An estimate of the number of non-zero values of local sparse matrix.
Scope: **local**
Type: **optional**
Default: $nnz = 9 * decomp\_data\%matrix\_data[n\_row]$.
Specified as: an integer variable.

On Return

a allocated local portion of global sparse matrix A.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

ierrv error vector used in subroutines F90_PSSPINS and F90_PSSPASB.
Scope: **local**
Type: **required**
Specified as: a one dimensional array of integer.

F90_PSSPINS—Insert Sparse Matrix

This subroutine inserts a local sparse matrix $BLCK$ in the global sparse matrix A.

$A, BLCK$	**Subroutine**
Long Precision Real	F90_PSSPINS
Long Precision Complex	F90_PSSPINS

Table 3.2: Data types

Syntax

CALL F90_PSSPINS (*a, ia, ja, blck, ierrv, decomp_data*)

CALL F90_PSSPINS
(*a, ia, ja, blck, ierrv, decomp_data, IBLCK=iblck, JBLCK=jblck*)

On Entry

a the local portion of global sparse matrix A. This subroutine computes the location of the first element of the local submatrix to insert, based on ia, ja and $matrix\_data$ field of *decomp_data*.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

ia the row index of global sparse matrix A, identifying the first row of the submatrix A.
Scope: **global**
Type: **required**
Specified as: an integer variable $1 \leq ia \leq decomp\_data\%matrix\_data[M\_]$.

ja the column index of the global sparse matrix A, identifying the first column of the submatrix A.
Scope: **global**
Type: **required**
Specified as: an integer variable $1 \leq ja \leq decomp\_data\%matrix\_data[N\_]$.

blck the local submatrix to insert.
Scope: **local**
Type: **required**

Specified as: a structured data type specified in § 1; supported storage formats are `CSR` and `COO`..

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

ierrv error vector. It must be initialized by F90_PSSPALL.
Scope: **local**
Type: **required**
Specified as: a one dimensional array of integer.

iblck the row index of local sparse matrix $BLCK$, identifying the first row of the submatrix $BLCK$.
Scope: **local**
Type: **optional**
Default: $iblck = 1$.
Specified as: an integer variable $jblck \geq 1$.

jblck the column index of the local sparse matrix $BLCK$, identifying the first column of the submatrix $BLCK$.
Scope: **local**
Type: **optional**
Default: $jblck = 1$.
Specified as: an integer variable $jblck \geq 1$.

On Return

a the local portion of global sparse matrix A updated with matrix $BLCK$.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

ierrv error vector used in successive call to F90_PSSPINS and F90_PSSPASB.
Scope: **local**
Type: **required**
Specified as: a one dimensional array of integer.

Usage Notes

1. Before call this subroutine you must allocate A with F90_PSSPALL subroutine.

2. IA, JA and the indices in $BLCK$ are written in terms of the "global" numbering; the mapping to local storage is handled internally.

3. The “insertion” in the routine name means that coefficients from the $BLCK$ parameter are stored into the global sparse matrix A starting from $A(IA, JA)$, i.e., element $BLCK(1, 1)$ (if it actually exists), goes onto $A(IA, JA)$, and so on. This does *not* imply that previously existing coefficients are overwritten; any clashes are resolved at assembly time (see subroutine F90_PSSPASB).

4. This routine does not contain any syncronization point; thus, it can be called independently by each of the participating processes; it is however required that all processes execute later a single call to F90_PSSPASB.

5. It is perfectly legal to call this routine multiple times with the same value of IA; this can be useful e.g. in finite element applications, where the most natural way to define the matrix is to loop over the elements, and not over the equations. As an extreme example, it is possible to build a sparse matrix passing just one nonzero entry to each call. However it is generally convenient from a performance point of view to pack data spanning multiple rows into a single call to this subroutine; the optimal size of each $BLCK$ of data depends on the underlying processor configuration.

F90_PSSPASB—Assembly Sparse Matrix

This subroutine assemblies global sparse matrix A.

A	**Subroutine**
Long Precision Real	F90_PSSPASB
Long Precision Complex	F90_PSSPASB

Table 3.3: Data types

Syntax

CALL F90_PSSPASB (*a, ierrv, decomp_data*)

CALL F90_PSSPASB
(*a, ierrv, decomp_data, OVERLAP=overlap, AFMT=afmt,UP=up,DUP=dup*)

On Entry

a the local portion of global sparse matrix A to assembly.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

ierrv error vector. It must be initialized by F90_PSSPALL and must be the same used in F90_PSSPINS.
Scope: **local**
Type: **required**
Specified as: a one dimensional array of integer.

overlap Permits to improve F90_PSSPASB performance, if there isn't overlap between local portions of global sparse matrix A assigned over all processes.
Has the following meaning:
If $overlap = 0$, there isn't overlap between local portions of global sparse matrix A assigned over all processes.
If $overlap \neq 0$, there is overlap between local portions of global sparse

matrix A assigned over all processes.
Scope: **global**
Type: **optional**
Default: $overlap = 1$.
Specified as: an integer variable.

afmt Request matrix A to be converted into a specfic storage format; it must be supported by the underlying serial sparse BLAS. It is assumed that at least `CSR` should be supported; the default is `???`, which requests the default of the underlying serial spares BLAS. Scope: **global**
Type: **optional**
Default: $overlap =$ `???`.
Specified as: a `CHARACTER*5` variable.

UP Request matrix A to be prepared for efficient pattern reuse (see subroutine F90_PSSPUPD). This entails additional storage overhead. Possible values are `Y` and `N`. Scope: **global**
Type: **optional**
Default: $UP =$ `N`.
Specified as: a `CHARACTER` variable.

DUP Specifies how to treat duplicated matrix entries; see usage notes below.
Scope: **global**
Type: **optional**
Default: $dup = 1$.
Specified as: an integer variable.

On Return

a the local portion of global sparse matrix A assembled.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

Usage Notes

1. Before call this subroutine you must allocate A with F90_PSSPALL subroutine and insert all rows of matrix A with F90_PSSPINS.

2. It may happen that calls to `F90_PSSPINS` have inserted multiple values with the same indices, i.e. you may have duplicated coefficient entries. This subroutine provides three ways of coping with duplicated entries by specifiyng the relevant value into the DUP variable:

   1 Raise an error, duplicated coefficients should not exist;

   2 Ignore replications, keep one of the coefficients found;

   3 Sum the duplicated entries to obtain the final coefficient.

F90_PSPTINS—Insert Pattern

This subroutine handles updates to a communication descriptor based only on the sparsity pattern. This is useful in those cases where we are only actually using a connectivity graph.

$BLCK$	**Subroutine**
	F90_PSPTINS

Table 3.4: Data types

Syntax

CALL F90_PSPTINS (*ia, ja, blck, ierrv, decomp_data*)

On Entry

ia the row index of global sparse matrix pattern A, identifying the first row of the submatrix pattern A.
Scope: **global**
Type: **required**
Specified as: an integer variable $1 \leq ia \leq decomp\_data\%matrix\_data[M\_]$.

ja the column index of the global sparse matrix A, identifying the first column of the submatrix A.
Scope: **global**
Type: **required**
Specified as: an integer variable $1 \leq ja \leq decomp\_data\%matrix\_data[N\_]$.

blck the local submatrix pattern to insert.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1; supported storage formats are `CSR` and `COO`..

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

ierrv error vector. It must be initialized by F90_PSSPALL.
Scope: **local**
Type: **required**
Specified as: a one dimensional array of integer.

On Return

ierrv error vector used in successive call to F90_PSSPINS and F90_PSSPASB.
Scope: **local**
Type: **required**
Specified as: a one dimensional array of integer.

Usage Notes

1. Before call this subroutine you must allocate $DECOMP\_DATA$ with F90_PSDSCALL subroutine.

2. IA, JA and the indices in $BLCK$ are written in terms of the "global" numbering; the mapping to local storage is handled internally.

3. This routine is very similar to F90_PSSPINS, but the coefficient values in the $BLCK$ parameter are completely ignored; only the nonzero pattern, i.e. the row and column indices, are used to update DECOMP_DATA.

4. This routine does not contain any syncronization point; thus, it can be called independently by each of the participating processes; it is however required that all processes execute later a single call to F90_PSPTASB.

5. It is perfectly legal to call this routine multiple times with the same value of IA; this can be useful e.g. in finite element applications, where the most natural way to define the matrix is to loop over the elements, and not over the equations. However it is generally convenient from a performance point of view to pack data spanning multiple rows into a single call to this subroutine; the optimal size of each $BLCK$ of data depends on the underlying processor configuration.

F90_PSPTASB—Assembly Sparse Communication Pattern

This subroutine assemblies a communication pattern DECOMP_DATA.

Subroutine
F90_PSPTASB

Table 3.5: Data types

Syntax

CALL F90_PSPTASB (*ierrv, decomp_data*)

On Entry

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

ierrv error vector. It must be initialized by F90_PSSPALL and must be the same used in F90_PSSPINS.
Scope: **local**
Type: **required**
Specified as: a one dimensional array of integer.

On Return

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

Usage Notes

1. Before call this subroutine you must allocate DECOMP_DATA with F90_PSDSCALL subroutine and insert all relevant pattern rows with F90_PSPTINS.

F90_PSSPFREE—Free Sparse Matrix

This subroutine deallocates sparse matrix's structures.

A	**Subroutine**
Long Precision Real	F90_PSSPFREE
Long Precision Complex	F90_PSSPFREE

Table 3.6: Data types

Syntax

CALL F90_PSSPFREE (*a*, *decomp_data*)

On Entry

a the local portion of global sparse matrix A to deallocate.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

decomp_data contains data structures for communications referred to sparse matrix to deallocate.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

On Return

a deallocated structure for storing other sparse matrices.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

decomp_data contains deallocated data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

Usage Notes

1. Before call this subroutine you must allocate A with F90_PSSPALL subroutine.

2. After this call, if no error encountered, data structures *a* and *decomp_data* can be used for storing other sparse matrices.

F90_PSDSCFREE—Free Communication Descriptor

This subroutine deallocates communication descriptors

Syntax

CALL F90_PSDSCFREE (*decomp_data*)

On Entry

decomp_data contains data structures for communications referred to sparse matrix to deallocate.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

On Return

decomp_data It is deallocated.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

Usage Notes

1. Before a call this subroutine you must have allocate *decomp_data* with F90_PSDSCALL subroutine.
2. After this call, if no error encountered, data structure *decomp_data* can be reused.

F90_PSDSALL—Allocate Global Dense Matrix

This subroutine allocates global dense matrix X.

X	**Subroutine**
Integer	F90_PSDSALL
Long Precision Real	F90_PSDSALL
Long Precision Complex	F90_PSDSALL

Table 3.7: Data types

Syntax

CALL F90_PSDSALL (*m, n, x, ierrv, decomp_data, js*)

CALL F90_PSDSALL (*m, x, ierrv, decomp_data*)

On Entry

m number of rows of global dense submatrix X to allocate.
Scope: **global**
Type: **required**
Specified as: an integer variable $0 \leq m \leq decomp\_data\%matrix\_data[N\_]$.

n number of columns of global dense submatrix X to allocate.
Scope: **global**
Type: **required**, but can only be present if x is of rank 2.
Specified as: an integer variable $n \geq 0$.

x a rank one or two array with the POINTER attribute.

ierrv see "On Return".

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

js starting index for the columns of global dense submatrix X to allocate.
Scope: **global**
Type: **optional**; can only be present if x is of rank 2.
Specified as: an integer variable; default: $js = 1$.

On Return

x allocated local portion of global dense matrix X.
Scope: **local**
Type: **required**
Specified as: a rank-one or two array containing numbers of the type indicated in Table 3.7, of size *decomp_data%matrix_data*[$N\_$] by n; the column indices are declared as $js : js + n - 1$.

ierrv error vector used in subroutines F90_PSDSINS and F90_PSDSASB.
Scope: **local**
Type: **required**
Specified as: a one dimensional array of integer.

Usage Notes

1. Before call This subroutine you must initialize *decomp_data* fields with F90_PSSPASB subroutine.

F90_PSDSINS—Insert Dense Matrix

This subroutine inserts a local dense matrix $BLCK$ in the global dense matrix X.

$X, BLCK$	**Subroutine**
Integer	F90_PSDSINS
Long Precision Real	F90_PSDSINS
Long Precision Complex	F90_PSDSINS

Table 3.8: Data types

Syntax

CALL F90_PSDSINS (*m, n, x, ix, jx, blck, ierrv, decomp_data*)

CALL F90_PSDSINS
(*m, n, x, ix, jx, blck, ierrv, decomp_data IBLCK=iblck, JBLCK=jblck*)

CALL F90_PSDSINS (*m, x, ix, blck, ierrv, decomp_data*)

On Entry

m number of rows of local dense submatrix $BLCK$ to insert.
Scope: **global**
Type: **required**
Specified as: an integer variable $0 \le m \le decomp\_data\%matrix\_data[M\_]$.

n number of columns of global dense submatrix $BLCK$ to insert.
Scope: **global**
Type: **required** but only if x is of rank 2.
Specified as: an integer variable .

x the local portion of global dense matrix X. This subroutine computes the location of the first element of the local subarray to insert, based on ix, jx and $matrix\_data$ field of *decomp_data*.
Scope: **local**
Type: **required**
Specified as: a rank one or two array with the POINTER attribute containing numbers of the type indicated in Table 3.8. The rank of *blck* must be the same of x.

ix the row index of global dense matrix X, identifying the first row of the submatrix X.
Scope: **global**
Type: **required**
Specified as: an integer variable $1 \leq ix \leq decomp\_data\%matrix\_data[N\_]$.

jx the column index of the global dense matrix X, identifying the first column of the submatrix X.
Scope: **global**
Type: **required**
Specified as: an integer variable $jx \geq 1$.

blck the local submatrix to insert.
Scope: **local**
Type: **required**
Specified as: a rank one or rank two containing numbers of the type indicated in Table 3.8. The rank of *blck* must be the same of x.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

ierrv error vector. It must be initialized by F90_PSDSALL.
Scope: **local**
Type: **required**
Specified as: a one dimensional array of integer.

iblck the row index of local dense matrix $BLCK$, identifying the first row of the submatrix $BLCK$.
Scope: **local**
Type: **optional**
Default: $iblck = 1$.
Specified as: an integer variable $jblck \geq 1$.

jblck the column index of the local dense matrix $BLCK$, identifying the first column of the submatrix $BLCK$.
Scope: **local**
Type: **optional**
Default: $jblck = 1$.
Specified as: an integer variable $jblck \geq 1$.

On Return

x the local portion of global dense matrix X. This subroutine computes the location of the first element of the local subarray to insert, based on ix, jx and $matrix\_data$ field of *decomp_data*.
Scope: **local**
Type: **required**

Specified as: a two dimensional array containing numbers of the type indicated in Table 3.8.

ierrv error vector used in successive call to F90_PSDSINS and F90_DASB.
Scope: **local**
Type: **required**
Specified as: a one dimensional array of integer.

Usage Notes

1. Before call this subroutine you must allocate X with F90_PSDSALL subroutine.

F90_PSDSASB—Assembly Dense Matrix

This subroutine assemblies global dense matrix X.

X	**Subroutine**
Integer	F90_PSDSASB
Long Precision Real	F90_PSDSASB
Long Precision Complex	F90_PSDSASB

Table 3.9: Data types

Syntax

CALL F90_PSDSASB (*x, ierrv, decomp_data*)

On Entry

x the local portion of global dense matrix X.
Scope: **local**
Type: **required**
Specified as: a rank one or two array with the POINTER attribute containing numbers of the type indicated in Table 3.9.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

ierrv error vector. It must be initialized by F90_PSDSALL and must be the same used in F90_PSDSINS.
Scope: **local**
Type: **required**
Specified as: a one dimensional array of integer. must be the same used in F90_PSDSINS.
Scope: **local**
Type: **optional**
Specified as: a one dimensional array of integer.

On Return

x the local portion of global dense matrix X.
Scope: **local**
Type: **required**

Specified as: a rank one or two array with the POINTER attribute containing numbers of the type indicated in Table 3.8.

ierrv error vector used in successive call to F90_PSDSINS and F90_PSDSASB.
Scope: **local**
Type: **required**
Specified as: a one dimensional array of integer.

Usage Notes

1. Before call this subroutine you must allocate X with F90_PSDSALL subroutine.

F90_PSDSFREE—Free Dense Matrix

These subroutine deallocates dense matrix's structures.

A	**Subroutine**
Integer	F90_PSDSFREE
Long Precision Real	F90_PSDSFREE
Long Precision Complex	F90_PSDSFREE

Table 3.10: Data types

Syntax

CALL F90_PSDSFREE (*X, decomp_data*)

On Entry

x the local portion of global dense matrix X.
Scope: **local**
Type: **required**
Specified as: a rank one or two array with the POINTER attribute containing numbers of the type indicated in Table 3.9.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

On Return

x deallocated local portion of global dense matrix X.
Scope: **local**
Type: **required**
Specified as: a two dimensional array containing numbers of the type indicated in Table 3.10

Usage Notes

1. Before call this subroutine you must allocate x with F90_PSDSALL subroutine.
2. These procedures cannot be executed if related sparse matrix descriptor has been already deallocated with F90_PSDSCFREE call.

F90_PSSPREINIT —Reinitialize Global Sparse Matrix

This subroutine reinitializes global sparse matrix A; the user planning to regenerate a matrix with exactly the same nonzero pattern should call this routine, then `F90_PSSPUPD` instead of `F90_PSSPINS`.

A	**Subroutine**
Long Precision Real	F90_PSSPREINIT
Long Precision Complex	F90_PSSPREINIT

Table 3.11: Data types

Syntax

CALL F90_PSSPREINIT (*a, ierrv, decomp_data*)

On Entry

a see "On Return".

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

ierrv see "On Return".

On Return

a allocated local portion of global sparse matrix A.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

ierrv error vector used in subroutines F90_PSSPUPD and F90_PSSPASB.
Scope: **local**
Type: **required**
Specified as: a one dimensional array of integer.

F90_PSSPUPD—Update Sparse Matrix

This subroutine updates the sparse matrix A with the local sparse matrix $BLCK$. This assumes that there has been a previous call to F90_PSSPASB with argument $UP = Y$. Moreover, for efficiency reasons, it is assumed that the block entries in BLCK are generated in exactly the same order as in the previous run using F90_PSSPINS.

$A, BLCK$	**Subroutine**
Long Precision Real	F90_PSSPUPD
Long Precision Complex	F90_PSSPUPD

Table 3.12: Data types

Syntax

CALL F90_PSSPUPD (*a, ia, ja, blck, ierrv, decomp_data*)

CALL F90_PSSPUPD
(*a, ia, ja, blck, ierrv, decomp_data, IBLCK=iblck, JBLCK=jblck*)

On Entry

a the local portion of global sparse matrix A. This subroutine computes the location of the first element of the local submatrix to insert, based on ia, ja and *matrix_data* field of *decomp_data*.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

ia the row index of global sparse matrix A, identifying the first row of the submatrix A.
Scope: **global**
Type: **required**
Specified as: an integer variable $1 \leq ia \leq decomp\_data\%matrix\_data[M\_]$.

ja the column index of the global sparse matrix A, identifying the first column of the submatrix A.
Scope: **global**
Type: **required**
Specified as: an integer variable $1 \leq ja \leq decomp\_data\%matrix\_data[N\_]$.

blck the local submatrix to insert.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1; supported storage formats are: `CSR`.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

ierrv error vector. It must be initialized by F90_PSSPALL.
Scope: **local**
Type: **required**
Specified as: a one dimensional array of integer.

iblck the row index of local sparse matrix $BLCK$, identifying the first row of the submatrix $BLCK$.
Scope: **local**
Type: **optional**
Default: $iblck = 1$.
Specified as: an integer variable $jblck \geq 1$.

jblck the column index of the local sparse matrix $BLCK$, identifying the first column of the submatrix $BLCK$.
Scope: **local**
Type: **optional**
Default: $jblck = 1$.
Specified as: an integer variable $jblck \geq 1$.

On Return

a the local portion of global sparse matrix A updated with matrix $BLCK$.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

ierrv error vector used in successive call to F90_PSSPINS and F90_PSSPASB.
Scope: **local**
Type: **required**
Specified as: a one dimensional array of integer.

Usage Notes

1. Before call this subroutine you must reinitialize A with F90_PSSPREINIT subroutine.

2. This Routine can be called a number of different times over all processes because doesn't contain any syncronization points. If an error is encountered, it will be displayed in psdspasb routine.

F90_PSCSRP—Right Perumtation of Sparse Matrix

This subroutine updates the sparse matrix A with the permutation matrix P or its transpose P^T, as:

$$A \leftarrow A \cdot P$$

A	**Subroutine**
Long Precision Real	F90_PSCSRP
Long Precision Complex	F90_PSCSRP

Table 3.13: Data types

Syntax

CALL F90_PSCSRP (*trans, iperm, a, decomp_data*)

On Entry

trans Whether the permutation or its transpose should be used.
Scope: **global**
Type: **required**
Specified as: a character variable `N` or `T`.

iperm The permutation to be applied, stored in vector form.
Scope: **global**
Type: **required**
Specified as: a rank one integer array.

a the local portion of global sparse matrix A.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

On Return

a the local portion of global sparse matrix A, with its column indices updated according to the specified permutation.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

Usage Notes

1. The descriptors must be assembled before calls to this subroutine.
2. See also `F90_PSDSCREN` for a usage example.

F90_PSGELP—Left Permutation of Dense Matrix

This subroutine updates the dense matrix X with the permutation matrix P or its transpose P^T, as:

$$X \leftarrow P \cdot X$$

X	**Subroutine**
Long Precision Real	F90_PSGELP
Long Precision Complex	F90_PSGELP

Table 3.14: Data types

Syntax

CALL F90_PSGELP (*trans, iperm, x, decomp_data*)

On Entry

trans Whether the permutation or its transpose should be used.
Scope: **global**
Type: **required**
Specified as: a character variable `N` or `T`.

iperm The permutation to be applied, stored in vector form.
Scope: **global**
Type: **required**
Specified as: a rank one integer array.

x the local portion of global dense matrix X.
Scope: **local**
Type: **required**
Specified as: a rank one or two array of data type specified in table 3.14.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

On Return

x the local portion of global dense matrix X.
Scope: **local**
Type: **required**
Specified as: a rank one or two array of data type specified in table 3.14.

Usage Notes

1. The descriptors must be assembled before calls to this subroutine.
2. See also `F90_PSDSCREN` for a usage example.

F90_PSDSCREN—Renumbering of Communication Descriptors

This subroutine updates the communication descriptor *decomp_data* according to the permutation matrix P or its transpose P^T.

Syntax

CALL F90_PSDSCREN (*trans, iperm, decomp_data*)

On Entry

trans Whether the permutation or its transpose should be used.
Scope: **global**
Type: **required**
Specified as: a character variable `N` or `T`.

iperm The permutation to be applied, stored in vector form.
Scope: **global**
Type: **required**
Specified as: a rank one integer array.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

On Return

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

Usage Notes

1. The descriptors must be assembled before calls to this subroutine.
2. The permutation routines are defined to handle item renumbering that may come about because of local data storage issues. If the particular data structure chosen for the sparse matrix A entails a row permutation stored in `A%PL`, each application of the matrix-vector product $y \leftarrow Ax$ may be slowed down by the need to apply it to recover the natural row ordering in the result vector y. The following code fragment shows how

to unwind the permutation upon entering the iteration loop and how to restore natural ordering upon exit:

```
do_renum_left = (a%pl(1) /= 0)
if (do_renum_left) then
   ipnull(1) = 0
   do i=1, n_row
     iperm(i)  = a%pl(i)
   enddo
   do i=n_row+1,n_col
     iperm(i)  = i
   enddo
   ipsave => a%pl
   a%pl   => ipnull
   call f90_psdscren('T',iperm,decomp_data)
   call f90_pscsrp('N',iperm,a,decomp_data)
   call f90_psgelp('T',iperm,x,decomp_data)
   call f90_psgelp('T',iperm,b,decomp_data)
endif
do while(convergence)
 ......
end do
if (do_renum_left) then
  call f90_psdscren('N',iperm,decomp_data)
  call f90_pscsrp('T',iperm,a,decomp_data)
  call f90_psgelp('N',iperm,x,decomp_data)
  call f90_psgelp('N',iperm,b,decomp_data)
  a%pl  => ipsave
endif
```

F90_PSVERIFY—Verify correctness of communication descriptor

This subroutine checks the local and global correctness of the communication descriptor, possibly reordering the communication patterns to avoid deadlocks. It also verifies the correctness of the matrix structure.

Syntax

CALL F90_PSVERIFY (*d,a,decomp_data,check_descr,convert_descr, h,decomp_data_out,work,trans,unitd*)

On Entry

d The main diagonal of the matrix used for scaling.
Scope: **local**
Type: **required**
Specified as: a rank one real array.

a the local portion of global sparse matrix A.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

check_descr contains specifics about checks to be performed.
Scope: **global**
Type: **required**
Specified as: an integer array of rank one.

work a work area
Scope: **local** Type: **required** Specified as: a rank one integer array.

trans Whether the permutation or its transpose should be used.
Scope: **global**
Type: **optional**
Specified as: a character variable `N` or `T`.

unitd specifies whether the diagonal matrix is unit or whether row or column scaling has to be performed.
Scope: **global**
Type: **optional**
Specified as: a character variable `U`, `L`, `R` or `B`.

On Return

convert_descr specifies what operations have been performed.
Scope: **global**
Type: **required**
Specified as: a rank one integer array.

h the eventually converted local portion of global sparse matrix .
Scope: **local**
Type: **required** Specified as: a structured data type specified in § 1.

decomp_data_out contains data structures for communications eventually modified.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

Usage Notes

1. The descriptors must be assembled before calls to this subroutine.
2. Different kind of checks can be performed depending on values supplied with check_descr parameter:
   - 0-th bit : if 1 then the communication descriptor is checked for deadlocks.
   - 1-th bit : if 1 then the local matrix is checked for correctness
   - 2-th bit : if 1 then a local consistency check of the communication descriptor is performed.
   - 4-th bit : if 1 then a global consistency check of the communication descriptor is performed.
3. Operation performed by this routine are specified in the convert_descr output argument:

   If PSVERIFY has found a deadlock in communication described in DESC_HALO list, then 0-th bit value is set to 1, 0 otherwise. If PSVERIFY has found deadlock in communication described in DESC_OVRLAP list, then 1-th bit value is set to 1, 0 otherwise. If PSVERIFY has converted input matrix A to a representation supported by PSBLAS then 1-th bit value is set to 1, 0 otherwise.

4. Updated values are provided within H and decomp_data_out output parameters if specified in convert_descr parameter.

PSGLOB_TO_LOC—Convert global to local numbering

This subroutine accepts an integer vector of arbitrary size containing indices in the global numbering scheme, and converts them into local numbering scheme.

Syntax

CALL PSGLOB_TO_LOC (*x,decomp_data,act*)

CALL PSGLOB_TO_LOC (*x,y,decomp_data,act*)

On Entry

x An integer vector of indices to be converted.
Scope: **local**
Type: **required**
Specified as: a rank one integer array.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

act specifies action to be taken in case of range errors. Scope: **global**
Type: **optional**
Specified as: a character variable `E`, `W` or `A`.

On Return

x If y is not present, then x is overwritten with the translated integer indices.
Scope: **global**
Type: **required**
Specified as: a rank one integer array.

y If y is not present, then y is overwritten with the translated integer indices, and x is left unchanged. Scope: **global**
Type: **optional**
Specified as: a rank one integer array.

Usage Notes

1. The descriptors must be assembled before calls to this subroutine.
2. The subroutine will convert the integer values contained in the input vector, mapping from the global numbering to local numbering.
3. The value of `ACT` will determine whether any range errors in the input vector will be treated as fatal errors, warnings or will be ignored. Possible errors are:

   - Value $x(i)$ is outside the global numbering range $1 \ldots decomp\_data\%matrix\_data(M\_)$ for some i;
   - Value $x(i)$ is within the global numbering range $1 \ldots decomp\_data\%matrix\_data(M\_)$, but there is no corresponding local index to be mapped onto;

4. A global index will have a corresponding local index if either:

   (a) The global index is assigned to the current process by the `PARTS` user defined routine;

   (b) The global index is in the halo, i.e. there is at least one nonzero entry in the local part of the matrix a corresponding to the descriptor *decomp_data* with the same column index.

PSLOC_TO_GLOB—Convert local to global numbering

This subroutine accepts an integer vector of arbitrary size containing indices in the local numbering scheme, and converts them into global numbering scheme.

Syntax

CALL PSLOC_TO_GLOB (*x,decomp_data,act*)

CALL PSLOC_TO_GLOB (*x,y,decomp_data,act*)

On Entry

x An integer vector of indices to be converted.
Scope: **local**
Type: **required**
Specified as: a rank one integer array.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

act specifies action to be taken in case of range errors. Scope: **global**
Type: **optional**
Specified as: a character variable `E`, `W` or `A`.

On Return

x If y is not present, then x is overwritten with the translated integer indices.
Scope: **global**
Type: **required**
Specified as: a rank one integer array.

y If y is not present, then y is overwritten with the translated integer indices, and x is left unchanged. Scope: **global**
Type: **optional**
Specified as: a rank one integer array.

Usage Notes

1. The descriptors must be assembled before calls to this subroutine.
2. The subroutine will convert the integer values contained in the input vector, mapping from the local numbering to global numbering.
3. The value of `ACT` will determine whether any range errors in the input vector will be treated as fatal errors, warnings or will be ignored. Possible errors are:
   - Value $x(i)$ is outside the local numbering range $1 \ldots decomp\_data\%matrix\_data(n\_col\_)$ for some i;
4. A global index will have a corresponding local index if either:
   (a) The global index is assigned to the current process by the `PARTS` user defined routine;
   (b) The global index is in the halo, i.e. there is at least one nonzero entry in the local part of the matrix a corresponding to the descriptor *decomp_data* with the same column index.

Chapter 4

Method Library

In this chapter we provide routines for preconditioners and iterative methods. Their interfaces are defined in the module F90METHD

PRECONDITIONER —Compute Preconditioner

This subroutine computes a preconditioner for sparse matrix A.

Syntax

CALL PRECONDITIONER (*iprec,a,l,u,vdiag,decomp_data,ierr*)

On Entry

iprec Type of preconditioner to be computed.
Scope: **global**
Type: **required**
Specified as: an integer variable:

- **0** No preconditioner;
- **1** Diagonal scaling;
- **2** Local $ILU(0)$ preconditioning;

a the local portion of global sparse matrix A.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

On Return

ierr error code.
Scope: **local**
Type: **required**
Specified as: an integer number.

l the local portion of global sparse matrix L of the $ILU(0)$ factorization of A
Scope: **local**
Type: **required**; it is only used when $iprec = 2$.
Specified as: a structured data type specified in § 1.

u the local portion of global sparse matrix U of the $ILU(0)$ factorization of A

Scope: **local**

Type: **required**; it is only used when $iprec = 2$.

Specified as: a structured data type specified in § 1. Specified as: a structured data type specified in § 1.

vdiag a diagonal scaling vector.

Scope: **local**

Type: **required**; it is used when $iprec = 2$ or $iprec = 1$.

Specified as: a rank one array with the POINTER attribute; if $iprec = 1$ it contains the diagonal scaling matrix; if $iprec = 2$ it contains the diagonal of the $ILU(0)$ factorization.

Usage Notes

1. This routine allocates the internal components of L and U as necessary; between two successive calls to this routine, the user should place a call to `F90_PSSPFREE` for both L and U to avoid memory leaks.

F90_CGS —CGS Iterative Method

This subroutine implements the CGS method with restarting. The stopping criterion is the normwise backward error, in the infinity norm, i.e. the iteration is stopped when

$$\frac{\|r\|}{(\|A\|\|x\| + \|b\|)} < eps$$

Syntax

CALL F90_CGS (*a,iprec,l,u,vdiag,b,x,eps,decomp_data*)

CALL F90_CGS
(*a,iprec,l,u,vdiag,b,x,eps, decomp_data,itmax,iter,err,ierr,itrace*)

On Entry

iprec Type of preconditioner to be computed.
Scope: **global**
Type: **required**
Specified as: an integer variable:

0 No preconditioner;

1 Diagonal scaling;

2 Local $ILU(0)$ preconditioning;

a the local portion of global sparse matrix A.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

l the local portion of global sparse matrix L of the $ILU(0)$ factorization of A, as returned by the `PRECONDITIONER` routine.
Scope: **local**
Type: **required**; it is only used when $iprec = 2$.
Specified as: a structured data type specified in § 1.

u the local portion of global sparse matrix U of the $ILU(0)$ factorization of A, as returned by the `PRECONDITIONER` routine.
Scope: **local**
Type: **required**; it is only used when $iprec = 2$.
Specified as: a structured data type specified in § 1. Specified as: a structured data type specified in § 1.

vdiag a diagonal scaling vector.
Scope: **local**
Type: **required**; it is used when $iprec = 2$ or $iprec = 1$.
Specified as: a rank one array , as returned by the `PRECONDITIONER` routine.

b The RHS vector.
Scope: **local**
Type: **required**
Specified as: a rank one array.

x The initial guess.
Scope: **local**
Type: **required**
Specified as: a rank one array.

eps The stopping tolerance.
Scope: **global**
Type: **required**
Specified as: a real number.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

itmax The maximum number of iterations to perform.
Scope: **global**
Type: **optional**
Default: $itmax = 1000$.
Specified as: an integer variable $itmax \geq 1$.

itrace A tracing parameter.
Scope: **global**
Type: **optional**
Default: $itrace = -1$.
Specified as: an integer variable $-1 \leq itrace \leq 99$; if $itrace > -1$, it is interpreted as a FORTRAN unit number, and is used to print a summary of the convergence parameters at each iteration.

On Return

x The computed solution.
Scope: **local**
Type: **required**
Specified as: a rank one array.

iter The number of iterations performed.
Scope: **global**

Type: **optional**
Returned as: an integer variable.

err The error estimate on exit.
Scope: **global**
Type: **optional**
Returned as: a real number.

ierr An error code.
Scope: **global**
Type: **optional**
Returned as: an integer variable:

-1 Invalid preconditioner;

0 Normal termination;

> 0 Method did not converge; set to the number of iterations performed.

F90_BICG —BiCG Iterative Method

This subroutine implements the BiCG method with restarting. The stopping criterion is the normwise backward error, in the infinity norm, i.e. the iteration is stopped when

$$\frac{\|r\|}{(\|A\|\|x\| + \|b\|)} < eps$$

Syntax

CALL F90_BICG (*a,iprec,l,u,vdiag,b,x,eps,decomp_data*)

CALL F90_BICG
(*a,iprec,l,u,vdiag,b,x,eps, decomp_data,itmax,iter,err,ierr,itrace*)

On Entry

iprec Type of preconditioner to be computed.
Scope: **global**
Type: **required**
Specified as: an integer variable:

0 No preconditioner;

1 Diagonal scaling;

2 Local $ILU(0)$ preconditioning;

a the local portion of global sparse matrix A.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

l the local portion of global sparse matrix L of the $ILU(0)$ factorization of A, as returned by the `PRECONDITIONER` routine.
Scope: **local**
Type: **required**; it is only used when $iprec = 2$.
Specified as: a structured data type specified in § 1.

u the local portion of global sparse matrix U of the $ILU(0)$ factorization of A, as returned by the `PRECONDITIONER` routine.
Scope: **local**
Type: **required**; it is only used when $iprec = 2$.
Specified as: a structured data type specified in § 1. Specified as: a structured data type specified in § 1.

vdiag a diagonal scaling vector.
Scope: **local**
Type: **required**; it is used when $iprec = 2$ or $iprec = 1$.
Specified as: a rank one array , as returned by the `PRECONDITIONER` routine.

b The RHS vector.
Scope: **local**
Type: **required**
Specified as: a rank one array.

x The initial guess.
Scope: **local**
Type: **required**
Specified as: a rank one array.

eps The stopping tolerance.
Scope: **global**
Type: **required**
Specified as: a real number.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

itmax The maximum number of iterations to perform.
Scope: **global**
Type: **optional**
Default: $itmax = 1000$.
Specified as: an integer variable $itmax \geq 1$.

itrace A tracing parameter.
Scope: **global**
Type: **optional**
Default: $itrace = -1$.
Specified as: an integer variable $-1 \leq itrace \leq 99$; if $itrace > -1$, it is interpreted as a FORTRAN unit number, and is used to print a summary of the convergence parameters at each iteration.

On Return

x The computed solution.
Scope: **local**
Type: **required**
Specified as: a rank one array.

iter The number of iterations performed.
Scope: **global**

Type: **optional**
Returned as: an integer variable.

err The error estimate on exit.
Scope: **global**
Type: **optional**
Returned as: a real number.

ierr An error code.
Scope: **global**
Type: **optional**
Returned as: an integer variable:

-1 Invalid preconditioner;

0 Normal termination;

> 0 Method did not converge; set to the number of iterations performed.

F90_BICGSTAB —BiCGSTAB Iterative Method

This subroutine implements the BiCGSTAB method with restarting. The stopping criterion is the normwise backward error, in the infinity norm, i.e. the iteration is stopped when

$$\frac{\|r\|}{(\|A\|\|x\| + \|b\|)} < eps$$

Syntax

CALL F90_BICGSTAB (*a,iprec,l,u,vdiag,b,x,eps,decomp_data*)

CALL F90_BICGSTAB
(*a,iprec,l,u,vdiag,b,x,eps, decomp_data,itmax,iter,err,ierr,itrace*)

On Entry

iprec Type of preconditioner to be computed.
Scope: **global**
Type: **required**
Specified as: an integer variable:

- **0** No preconditioner;
- **1** Diagonal scaling;
- **2** Local $ILU(0)$ preconditioning;

a the local portion of global sparse matrix A.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

l the local portion of global sparse matrix L of the $ILU(0)$ factorization of A, as returned by the `PRECONDITIONER` routine.
Scope: **local**
Type: **required**; it is only used when $iprec = 2$.
Specified as: a structured data type specified in § 1.

u the local portion of global sparse matrix U of the $ILU(0)$ factorization of A, as returned by the `PRECONDITIONER` routine.
Scope: **local**
Type: **required**; it is only used when $iprec = 2$.
Specified as: a structured data type specified in § 1. Specified as: a structured data type specified in § 1.

vdiag a diagonal scaling vector.
Scope: **local**
Type: **required**; it is used when $iprec = 2$ or $iprec = 1$.
Specified as: a rank one array , as returned by the `PRECONDITIONER` routine.

b The RHS vector.
Scope: **local**
Type: **required**
Specified as: a rank one array.

x The initial guess.
Scope: **local**
Type: **required**
Specified as: a rank one array.

eps The stopping tolerance.
Scope: **global**
Type: **required**
Specified as: a real number.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

itmax The maximum number of iterations to perform.
Scope: **global**
Type: **optional**
Default: $itmax = 1000$.
Specified as: an integer variable $itmax \geq 1$.

itrace A tracing parameter.
Scope: **global**
Type: **optional**
Default: $itrace = -1$.
Specified as: an integer variable $-1 \leq itrace \leq 99$; if $itrace > -1$, it is interpreted as a FORTRAN unit number, and is used to print a summary of the convergence parameters at each iteration.

On Return

x The computed solution.
Scope: **local**
Type: **required**
Specified as: a rank one array.

iter The number of iterations performed.
Scope: **global**
Type: **optional**
Returned as: an integer variable.

err The error estimate on exit.
Scope: **global**
Type: **optional**
Returned as: a real number.

ierr An error code.
Scope: **global**
Type: **optional**
Returned as: an integer variable:

-1 Invalid preconditioner;

0 Normal termination;

> 0 Method did not converge; set to the number of iterations performed.

F90_BICGSTABL —BiCGSTAB(L) Iterative Method

This subroutine implements the BiCGSTAB(L) method with restarting. The stopping criterion is the normwise backward error, in the infinity norm, i.e. the iteration is stopped when

$$\frac{\|r\|}{(\|A\|\|x\| + \|b\|)} < eps$$

Syntax

CALL F90_BICGSTABL (a,$iprec$,l,u,$vdiag$,b,x,eps,$decomp\_data$)

CALL F90_BICGSTABL
(a,$iprec$,l,u,$vdiag$,b,x,eps, $decomp\_data$,$itmax$,$iter$,err,$ierr$,$itrace$,ml)

On Entry

iprec Type of preconditioner to be computed.
Scope: **global**
Type: **required**
Specified as: an integer variable:

0 No preconditioner;

1 Diagonal scaling;

2 Local $ILU(0)$ preconditioning;

a the local portion of global sparse matrix A.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

l the local portion of global sparse matrix L of the $ILU(0)$ factorization of A, as returned by the `PRECONDITIONER` routine.
Scope: **local**
Type: **required**; it is only used when $iprec = 2$.
Specified as: a structured data type specified in § 1.

u the local portion of global sparse matrix U of the $ILU(0)$ factorization of A, as returned by the `PRECONDITIONER` routine.
Scope: **local**
Type: **required**; it is only used when $iprec = 2$.
Specified as: a structured data type specified in § 1. Specified as: a structured data type specified in § 1.

vdiag a diagonal scaling vector.
Scope: **local**
Type: **required**; it is used when $iprec = 2$ or $iprec = 1$.
Specified as: a rank one array , as returned by the `PRECONDITIONER` routine.

b The RHS vector.
Scope: **local**
Type: **required**
Specified as: a rank one array.

x The initial guess.
Scope: **local**
Type: **required**
Specified as: a rank one array.

eps The stopping tolerance.
Scope: **global**
Type: **required**
Specified as: a real number.

decomp_data contains data structures for communications.
Scope: **local**
Type: **required**
Specified as: a structured data type specified in § 1.

itmax The maximum number of iterations to perform.
Scope: **global**
Type: **optional**
Default: $itmax = 1000$.
Specified as: an integer variable $itmax \geq 1$.

itrace A tracing parameter.
Scope: **global**
Type: **optional**
Default: $itrace = -1$.
Specified as: an integer variable $-1 \leq itrace \leq 99$; if $itrace > -1$, it is interpreted as a FORTRAN unit number, and is used to print a summary of the convergence parameters at each iteration.

ml The (L) parameter.
Scope: **global**
Type: **optional**
Default: $ml = 1$.
Specified as: an integer variable $ml \geq 1$

On Return

x The computed solution.
Scope: **local**
Type: **required**
Specified as: a rank one array.

iter The number of iterations performed.
Scope: **global**
Type: **optional**
Returned as: an integer variable.

err The error estimate on exit.
Scope: **global**
Type: **optional**
Returned as: a real number.

ierr An error code.
Scope: **global**
Type: **optional**
Returned as: an integer variable:

-1 Invalid preconditioner;

0 Normal termination;

> 0 Method did not converge; set to the number of iterations performed.

Related Publications

1. Duff, I., Marrone, M., Radicati, G., and Vittoli, C. *Level 3 basic linear algebra subprograms for sparse matrices: a user level interface* ACM Trans. Math. Softw. 23, 3 (September), 379–401, 1997.

2. Filippone, S. and Colajanni, M., *PSBLAS: a library for parallel libear algebra computation on sparse matrices*, ACM Trans. Math. Softw. 26, 4 (December), 2000.