F11ZPFP

NAG Parallel Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check for implementation-dependent details. You are advised to enclose any calls to NAG Parallel Library routines between calls to Z01AAFP and Z01ABFP.

Note: you should read the the F11 Chapter Introduction before using this routine.

1 Description

F11ZPFP takes a coordinate storage representation of an n by n complex sparse matrix A, distributed in cyclic row block form, and reorders the non-zero entries on each processor such that subsequent operations involving the matrix A can be performed efficiently. Options are provided to handle the cases when entries have zero values or duplicated row and column indices.

F11ZPFP also initialises the array IAINFO, which is used by a number of F11 routines to store auxiliary information about the matrix A. Additionally, F11ZPFP transforms the global row and column indices of the coordinate storage representation to local indices as described in Section 2.6.1 of the F11 Chapter Introduction.

F11ZPFP must be called for each complex sparse matrix A, represented in distributed coordinate storage format and distributed in cyclic row block form, before any other library routine can be applied to A. The only exception are matrices A which have the same pattern of non-zero entries as a matrix previously set-up by a call to F11ZPFP with KIND = 'R'. For such matrices only the non-zero entries of A need to be reordered, and F11YNFP should be used to perform this task.

2 Specification

```
 SUBROUTINE F11ZPFP(ICNTXT, N, MB, NNZ, A, IROW, ICOL, DUP, ZERO,
1                   SYMM, KIND, IAINFO, LIA, IFAIL)
 COMPLEX*16         A(*)
 INTEGER            ICNTXT, N, MB, NNZ, IROW(*), ICOL(*),
1                   IAINFO(*), LIA, IFAIL
 CHARACTER*1        DUP, ZERO, SYMM, KIND
```

3 Usage

3.1 Definitions

The following definitions are used in describing the data distribution within this document:

m_p	–	the number of rows in the Library Grid.
n_p	–	the number of columns in the Library Grid.
p	–	$m_p \times n_p$, the total number of processors in the Library Grid.
p_r	–	the row grid coordinate of the calling processor.
p_c	–	the column grid coordinate of the calling processor.
r	–	$p_c + p_r n_p$, the rank of the calling processor according to the row major ordering of the Library Grid.
M_b	–	the blocking factor for the distribution of the rows of the matrix.

3.2 Global and Local Arguments

The following global **input** arguments must have the same value on entry to the routine on each processor and the global **output** arguments will have the same value on exit from the routine on each processor:

Global input arguments: N, MB, DUP, ZERO, SYMM, KIND, IFAIL

Global output arguments: IFAIL

The remaining arguments are local.

3.3 Distribution Strategy

Blocks of M_b contiguous rows of the matrix A are stored in coordinate storage format on the Library Grid cyclically row by row (i.e., in the row major ordering of the grid) starting from the {0,0} logical processor. This data distribution is described in more detail in Section 2.5 of the F11 Chapter Introduction.

This routine assumes that the data has already been correctly distributed, and if this is not the case will fail to produce correct results.

3.4 Related Routines

Some Library routines can be used to generate or distribute complex sparse matrices in cyclic row block form:

Complex sparse matrix generation: F01YPFP or F01YQFP

Complex sparse matrix distribution: F01XPFP

4 Arguments

1: ICNTXT — INTEGER *Local Input*

On entry: the Library context, usually returned by a call to the Library Grid initialisation routine Z01AAFP.

Note: the value of ICNTXT **must not** be changed.

2: N — INTEGER *Global Input*

On entry: the order n, of the matrix A.

Constraint: N ≥ 1.

3: MB — INTEGER *Global Input*

On entry: the blocking factor, M_b, used to distribute the rows of the matrix A.

Constraint: MB ≥ 1.

4: NNZ — INTEGER *Local Input/Local Output*

On entry: the number of non-zero entries of the matrix A stored on the calling processor.

Constraint: NNZ > 0.

On exit: the number of local non-zero entries with pairwise distinct row and column indices.

Because of removal of duplicates and zero entries, the value of NNZ on exit may be less than its value on entry.

5: A(∗) — COMPLEX*16 array *Local Input/Local Output*

Note: the dimension of the array A must be at least max(1,NNZ).

On entry: the non-zero entries in the blocks of the matrix A assigned to the calling processor. These may be in any order and there may be multiple non-zero entries with the same row and column indices.

On exit: the local non-zero entries reordered to enable subsequent operations involving the matrix A. Each local non-zero entry has a unique row and column index.

6: IROW(∗) — INTEGER array *Local Input/Local Output*

Note: the dimension of the array IROW must be at least max(1,NNZ).

On entry: the global row indices of the non-zero entries supplied in A.

Constraint: the elements of IROW must specify valid row indices of the blocks of the matrix A assigned to the calling processor, i.e., $1 \leq \mathrm{IROW}(l) \leq n$ and $\mathrm{mod}(\lfloor(\mathrm{IROW}(l)-1)/M_b\rfloor, p) = r$ for $l = 1, 2, \ldots,$NNZ. ($\lfloor x \rfloor$ denotes the integer part of a real number x.)

On exit: the first NNZ elements contain the local row indices of the non-zero entries returned in the array A. The contents of this array must not be changed between successive calls to library routines involving the matrix A.

7: ICOL(∗) — INTEGER array *Local Input/Local Output*

Note: the dimension of the array ICOL must be at least max(1,NNZ).

On entry: the global column indices of the non-zero entries supplied in A.

Constraint: $1 \leq \mathrm{ICOL}(l) \leq n$ for $l = 1, 2, \ldots,$NNZ.

On exit: the first NNZ elements contain the local column indices of the non-zero entries returned in the array A. The contents of this array must not be changed between successive calls to library routines involving the matrix A.

8: DUP — CHARACTER*1 *Global Input*

On entry: indicates how any non-zero entries with duplicate row and column indices are to be treated:

if DUP = 'R', the entries are removed;
if DUP = 'S', the relevant values in A are summed;
if DUP = 'F', the routine fails on detecting a duplicate, with IFAIL = 1 on exit.

Constraint: DUP = 'R', 'S', or 'F'.

9: ZERO — CHARACTER*1 *Global Input*

On entry: indicates how any entries with zero values in A are to be treated:

if ZERO = 'R', the entries are removed;
if ZERO = 'K', the entries are kept;
if ZERO = 'F', the routine fails on detecting a zero, with IFAIL = 2 on exit.

Constraint: ZERO = 'R', 'K', or 'F'.

10: SYMM — CHARACTER*1 *Global Input*

On entry: specifies whether the sparsity pattern (see Section 2.4.2 of the F11 Chapter Introduction) of the matrix A is symmetric or unsymmetric (see Section 6.1):

if SYMM = 'S', the sparsity pattern of the matrix A is symmetric;
if SYMM = 'U', the sparsity pattern of the matrix A is unsymmetric.

Note: no attempt is made to verify that the sparsity pattern of A is indeed symmetric or unsymmetric.

Constraint: SYMM = 'S' or 'U'.

11: KIND — CHARACTER*1 *Global Input*

On entry: indicates the relationship between the sparse matrix A and other sparse matrices to be dealt with subsequently:

if KIND = 'R', the sparsity pattern of A will be repeated in subsequent matrices;
if KIND = 'N', the sparsity pattern of A will not be repeated in subsequent matrices.

Constraint: KIND = 'R' or 'N'.

12: IAINFO(∗) — INTEGER array *Local Workspace/Local Output*

Note: the dimension of the array A must be at least max(200,LIA).

On exit: auxiliary information about the matrix A. IAINFO(1) contains the minimum required value of LIA. The first IAINFO(2) elements of IAINFO contain information required by other library routines and therefore must not be changed between successive calls to library routines involving the matrix A. See Section 3.3 of the F11 Chapter Introduction.

Note: if, on exit, IFAIL = 3, IAINFO(1) may be **smaller** than the minimum required value of LIA (when LIA > 0), and modifying the code by setting LIA to the returned value of IAINFO(1) may still cause failure; if, on exit, IFAIL = 4 (when LIA $= -1$), then an error occurred in the internal memory allocation mechanism.

13: LIA — INTEGER *Local Input*

Note: if LIA is set to -1, F11ZPFP will allocate the memory required for storing the auxiliary information about the matrix A (see Section 6.2). In this case, LIA is treated as a **global** input, i.e., LIA must be equal to -1 on all processors.

On entry: if LIA $\neq -1$, the dimension of the array IAINFO as declared in the (sub)program from which F11ZPFP is called.

Suggested value: LIA $= -1$ unless the required value of LIA was determined by a prior call to F11ZPFP.

Constraint: LIA $= -1$ or LIA ≥ 200.

14: IFAIL — INTEGER *Global Input/Global Output*

The NAG Parallel Library provides a mechanism, via the routine Z02EAFP, to reduce the amount of parameter validation performed by this routine. For a full description refer to the Z02 Chapter Introduction.

On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this argument (described in the Essential Introduction) the recommended values are:

IFAIL = 0, if multigridding is **not** employed;
IFAIL = -1, if multigridding is employed.

On exit: IFAIL = 0 (or -9999 if reduced error checking is enabled) unless the routine detects an error (see Section 5).

5 Errors and Warnings

If on entry IFAIL = 0 or -1, explanatory error messages are output from the root processor (or processor {0,0} when the root processor is not available) on the current error message unit (as defined by X04AAF).

5.1 Full Error Checking Mode Only

IFAIL = -2000

The routine has been called with an invalid value of ICNTXT on one or more processors.

IFAIL = -1000

The logical processor grid and library mechanism (Library Grid) have not been correctly defined, see Z01AAFP.

IFAIL = $-i$

On entry, the ith argument was invalid. This error occured either because a global argument did not have the same value on all logical processors, or because its value on one or more processors was incorrect. An explanatory message distinguishes between these two cases.

IFAIL = 3

LIA is too small, resulting in insufficient space to store the required auxiliary information in the array IAINFO.

5.2 Any Error Checking Mode

IFAIL = 1

DUP = 'F', and non-zero entries have been supplied which have duplicate row and column indices.

IFAIL = 2

ZERO = 'F', and at least one matrix entry has been supplied with a zero coefficient value.

IFAIL = 4

An error in the internal memory allocation mechanism occurred. This is usually due to insufficent memory resources.

6 Further Comments

6.1 Matrices with Symmetric Sparsity Pattern

If the sparsity pattern of the matrix A is symmetric, the routine can perform all operations on each logical processor independently. In contrast, setting up the necessary information requires communication between logical processors if the sparsity pattern of the matrix A is unsymmetric. Therefore, setting SYMM = 'S' for matrices with symmetric sparsity pattern can lead to a performance improvement.

6.2 Releasing Internally Allocated Resources

F11ZPFP will allocate the memory required for storing the auxiliary information about the matrix A if LIA is globally set to -1 on entry. When this information is no longer needed, the allocated memory should be released by a call to F11ZZFP.

7 References

None.

8 Example

See Section 8 of the document for F11BRFP.