F11ZAFP

NAG Parallel Library Routine Document

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

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.

1 Description

Note: F11ZAFP has been replaced by F11ZBFP, which should be called in preference to F11ZAFP. This routine will be withdrawn at the next release.

F11ZAFP takes a coordinate storage representation of an n by n real sparse matrix A, distributed on a logical grid of processors in cyclic row block form, and reorders the non-zero elements on each processor such that subsequent operations involving the matrix A can be performed efficiently. Entries with duplicate row and column indices may be removed, or their values may be summed. Any entries with zero values may optionally be removed.

F11ZAFP also initialises the array IAINFO, which is used by a number of F11 routines to store auxiliary information about the matrix A. Additionally, F11ZAFP 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.

F11ZAFP is a set-up routine which must be called for each real sparse matrix A, represented in distributed coordinate storage format, before any other library routine can be applied to A.

2 Specification

```
 SUBROUTINE F11ZAFP(ICNTXT, N, MB, NNZ, A, IROW, ICOL, DUP, ZERO,
1                   IAINFO, LIA, IFAIL)
 DOUBLE PRECISION   A(*)
 INTEGER            ICNTXT, N, MB, NNZ, IROW(*), ICOL(*),
1                   IAINFO(LIA), LIA, IFAIL
 CHARACTER*1        DUP, ZERO
```

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, 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 a logical grid of processors 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. However, the Library provides a utility routine, F01YAFP, which assists you in distributing the data correctly. A description of this routine can be found in Chapter F01 of the NAG Parallel Library

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 elements of the matrix A stored on the calling processor.

Constraint: NNZ > 0.

On exit: the number of local non-zero elements 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($*$) — DOUBLE PRECISION array *Local Input/Local Output*

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

On entry: the non-zero elements 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 elements with the same row and column indices.

On exit: the local non-zero elements reordered to enable subsequent operations involving the matrix A. Each local non-zero element 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 elements 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 elements 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 elements supplied in A.

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

On exit: the first NNZ elements contain the local column indices of the non-zero elements 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 elements 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 elements 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: IAINFO(LIA) — INTEGER array *Local Output*

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 = 0, IAINFO(1) contains the maximum number of elements of IAINFO used by F11ZAFP and thus provides the minimum required value of LIA. If IFAIL = 3, F11ZAFP attempted to use IAINFO(1) elements of IAINFO at a particular computational stage but found that LIA was too small to accomodate this storage requirement. In this case IAINFO(1) may be **smaller** than the minimum required value of LIA because more than IAINFO(1) elements might have been required to perform subsequent computational steps. Nevertheless, IAINFO(1) still gives a valuable indication of the minimum value of LIA to be used in future program runs.

11: LIA — INTEGER *Local Input*

On entry: the dimension of the array IAINFO as declared in the (sub)program from which F11ZAFP is called.

Suggested value: a value in the range $8m_l + 8n_{\text{LB}} + 2p + 100$ to $10m_l + 8n_{\text{LB}} + 2p + 100$ should be adequate for most problems. The quantities m_l and n_{LB} can be determined by $m_l =$ Z01CAFP(N,MB,r,0,p) and $n_{\text{LB}} = \lfloor(\lfloor(\text{N}-1/\text{MB}\rfloor + p - r)/p\rfloor$.

Constraint: LIA ≥ 200.

12: 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

5.1 Full Error Checking Mode Only

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).

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 had an invalid value. For global arguments, this may also be caused by an argument not having the same value on all logical processors. An explanatory message distinguishes between these two cases.

IFAIL = 1

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

IFAIL = 2

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

IFAIL = 3

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

6 Further Comments

6.1 Parallelism Detail

The routine performs all operations on each logical processor independently.

6.2 Computational Costs

The time taken for a call to F11ZAFP is approximately proportional to the maximum value of the number of non-zero elements, NNZ, on any processor.

7 References

[1] Saad Y (1996) *Iterative Methods for Sparse Linear Systems* PWS Publishing Company, Boston, MA

8 Example

None.