F11DDFP

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

F11DDFP applies a specified number of iterations of the forward, backward or symmetric SOR method to the real sparse system of linear equations:

$$Ax = y,$$

where A must have a **symmetric sparsity pattern** and is represented in coordinate storage format and distributed in cyclic row block form (see Sections 2.4.2, 2.5 and 2.7 of the F11 Chapter Introduction).

A call to F11DDFP must always be preceded by a call to F11ZBFP, to set up auxiliary information about A, and by a call to F11ZGFP to generate a multi-colour ordering of A.

It is envisaged that F11DDFP will be mostly used for the preconditioning step required in the application of F11BBFP or F11GBFP to sparse linear systems.

It is recommended that the user should read Section 6, before proceeding to use this routine for the first time.

2 Specification

```
 SUBROUTINE F11DDFP(ICNTXT, METH, NITS, N, NNZ, A, IROW, ICOL,
1                   INVDIA, RDIAG, OMEGA, Y, X, IAINFO, WORK, IFAIL)
 INTEGER            ICNTXT, NITS, N, NNZ, IROW(*), ICOL(*),
1                   IAINFO(*), IFAIL
 DOUBLE PRECISION   A(*), RDIAG(*), OMEGA, Y(*), X(*), WORK(*)
 CHARACTER*1        METH, INVDIA
```

3 Usage

3.1 Definitions

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

M_b	–	the blocking factor used in the cyclic row block distribution.
m_l	–	the number of rows of the matrix assigned to the calling processor (m_l = IAINFO(3), see IAINFO).
n_{int}^i	–	the number of internal interface indices (see Section 2.6.1 of the F11 Chapter Introduction) for the calling processor (n_{int}^i = IAINFO(6), see IAINFO).
n_{int}^e	–	the number of external interface indices (see Section 2.6.1 of the F11 Chapter Introduction) for the calling processor (n_{int}^e = IAINFO(7), see IAINFO).

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:	METH, NITS, N, INVDIA, OMEGA, IFAIL
Global output arguments:	IFAIL

The remaining arguments are local.

3.3 Distribution Strategy

The matrix A must be distributed in cyclic row block form.

When A is distributed in cyclic row block form, 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.

The vectors x and y are distributed conformally to the sparse matrix A, i.e., x and y are distributed across the Library Grid in the same way as each of the columns of the matrix A.

These data distributions are 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 real sparse matrices in cyclic row block form, or to generate or distributed vectors conformally to a given sparse matrix.

Real sparse matrix generation:	F01YAFP or F01YBFP
Real sparse matrix distribution:	F01XAFP
Real vector generation:	F01YEFP
Real vector scatter:	F01XEFP

Other library routines employ the SOR preconditioning:

Black Box routines:	F11DEFP for the solution of real general linear systems, F11JEFP for the solution of real symmetric systems.

3.5 Requisites

The sparse matrix A, which must have symmetric sparsity pattern, must have been preprocessed to set up the auxiliary information vector IAINFO by F11ZBFP.

The multi-colour ordering for F11DDFP must have been generated by calling F11ZGFP.

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: METH — CHARACTER*1 *Global Input*

On entry: specifies which SOR iterative method should be used:

if METH = 'F', for the forward SOR method;
if METH = 'B', for the backward SOR method;
if METH = 'S', for the symmetric SOR method.

Constraint: METH = 'F', 'B' or 'S'.

3: NITS — INTEGER *Global Input*

On entry: the number of iterations to be performed.

Constraint: NITS ≥ 1.

4: N — INTEGER *Global Input*

On entry: n, the order of the matrix A. It must contain the same value as the parameter N used in a prior call to F11ZBFP in which the array IAINFO was initialised.

Constraint: $\mathrm{N} \geq 1$.

5: NNZ — INTEGER *Local Input*

On entry: the number of non-zero entries in the matrix A stored on the calling processor. It must contain the same value as the parameter NNZ returned from a prior call to F11ZBFP in which the array IAINFO was initialised.

Constraint: $\mathrm{NNZ} > 0$.

6: A($*$) — DOUBLE PRECISION array *Local Input*

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. The local non-zero entries must have been reordered by prior calls to F11YAFP and F11ZBFP.

7: IROW($*$) — INTEGER array *Local Input*
8: ICOL($*$) — INTEGER array *Local Input*

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

On entry: the local row and column indices of the non-zero entries supplied in the array A. The contents of the arrays IROW and ICOL **must not** be changed between successive calls to library routines involving the matrix A.

Constraint: globally the IROW and ICOL indices must specify a symmetric sparsity pattern.

9: INVDIA — CHARACTER*1 *Global Input*

On entry: specifies whether the inverse values of the diagonal elements of A are provided explicitly:

if INVDIA = 'U', then the inverse values of the diagonal elements must be supplied in the array RDIAG;
if INVDIA = 'N', then the user does not require the inverse values of the diagonal elements to be returned in the array RDIAG;
if INVDIA = 'C', then the inverse values of the diagonal elements are calculated by F11DDFP and returned in the array RDIAG.

The provision of the inverse values of the diagonal elements eliminates the need to perform divisions in F11DDFP and thus can lead to some performance improvement.

Constraint: INVDIA = 'U', 'N' or 'C'.

10: RDIAG($*$) — DOUBLE PRECISION array *Local Input/Local Output*

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

On entry: if INVDIA = 'U', then RDIAG(i), for $i = 1, \ldots, m_l$, must contain the inverse, $1/a_{ii}$, of the ith diagonal element, according to the local indexing scheme, of A. Otherwise, the input values of the elements of RDIAG are not used.

On exit: if INVDIA = 'C', then RDIAG contains the inverse values of the diagonal elements calculated by F11DDFP. Otherwise, the elements of RDIAG are not changed.

11: OMEGA — DOUBLE PRECISION *Global Input*

On entry: the relaxation parameter ω.

Constraint: $0.0 < \mathrm{OMEGA} < 2.0$.

12: Y($*$) — DOUBLE PRECISION array *Local Input*

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

On entry: the local part of the vector y.

13: X($*$) — DOUBLE PRECISION array *Local Output*

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

On exit: the local part of the last iterate x_{NITS} (see Section 6.1).

14: IAINFO($*$) — INTEGER array *Local Input*

Note: the dimension of the array IAINFO must be at least max(200,IAINFO(2)).

On entry: the first IAINFO(2) elements of IAINFO contain auxiliary information about the matrix A. The array IAINFO must be initialised by a prior call to F11ZBFP and additional information must be stored in IAINFO by a prior call to F11ZGFP. The first IAINFO(2) elements of IAINFO must not be changed between successive calls to library routines involving the matrix A.

15: WORK($*$) — DOUBLE PRECISION array *Workspace*

Note: the dimension of the array WORK must be at least n^e_{int}+max(n^e_{int},n^i_{int}).

16: 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 = 1

IAINFO was not set up by a prior call to F11ZBFP followed by a call to F11ZGFP.

IFAIL = 2

On entry, the data stored in the arguments N, NNZ, IROW, ICOL and IAINFO is inconsistent. This indicates that, after the array IAINFO was set up by a call to F11ZBFP and F11ZGFP, at least one of these arguments was changed between successive calls to library routines.

IFAIL = 3

At least one diagonal element of A is zero.

6 Further Comments

6.1 Algorithmic Details

The iteration step performed by the SOR method is defined by

$$(D+\omega L)Px_{k+1} = [(1-\omega)D-\omega U]Px_k+\omega Py$$

for the forward SOR method,

$$(D+\omega U)Px_{k+1} = [(1-\omega)D-\omega L]Px_k+\omega Py,$$

for the backward SOR method and by

$$(D+\omega U)Px_{k+1} = [(1-\omega)D-\omega L](D+\omega L)^{-1}[(1-\omega)D-\omega U]Px_k+\omega(2-\omega)D(D+\omega L)^{-1}Py$$

for the symmetric SOR method. Above, the matrix A, of symmetric sparsity pattern, has been split into its strictly lower triangular, strictly upper triangular and diagonal parts L, U and D, respectively. The permutation matrix P is defined by the multi-colour ordering generated by a previous call to F11ZGFP. The starting vector x_0 is assumed to be zero in all cases.

7 References

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

8 Example

See Section 8 of the document for F11ZGFP.