F11DFFP

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

F11DFFP computes the incomplete LU factorizations of the local diagonal blocks of an n by n real sparse matrix A, represented in coordinate storage format (see Sections 2.6.1 and 2.7 of the F11 Chapter Introduction). The matrix A must have been distributed in cyclic row block form (see Sections 2.5.1 and 2.5.2 of the F11 Chapter Introduction). The matrix consisting of the factorized diagonal blocks can then be used for preconditioning by F11DHFP or F11BBFP.

The diagonal blocks A_k, for $k = 1, 2, \ldots, n_{\mathrm{LB}}$, on each processor, are decomposed as:

$$A_k = M_k + R_k$$

where

$$M_k = P_k L_k D_k U_k Q_k$$

and L_k is lower triangular with unit diagonal entries, D_k is diagonal, U_k is upper triangular with unit diagonal elements, P_k and Q_k are permutation matrices, and R_k is a remainder matrix.

The amount of fill-in occurring in each factorization can vary from no fill to complete fill, and can be controlled to some extent by specifying either the maximum level of fill using the argument LFILL, or the drop tolerance using the argument DTOL.

The argument PSTRAT defines the pivoting strategy to be used within each diagonal block. Four distinct pivoting strategies can be employed: no pivoting; user-defined pivoting; row partial pivoting by rows for stability; pivoting by columns for sparsity and by rows for stability. Optionally, a modified incomplete LU factorization (MILU) can be carried out which preserves the row-sums of the original diagonal blocks.

The diagonal blocks, M_k, of the preconditioning matrix are stored in CS format as the elements of the matrices

$$C_k = L_k + D_k^{-1} + U_k - 2I_k,$$

where I_k is the identity matrix. Additional auxiliary information is stored in the array IAINFO and will be used in subsequent calls to the preconditioning routine F11DGFP.

For further details see Section 6.

2 Specification

```
 SUBROUTINE F11DFFP(ICNTXT, N, NNZ, A, IROW, ICOL, NOVRLP, LFILL,
1                   DTOL, PSTRAT, MILU, IPIVP, IPIVQ, NNZC, C, LC,
2                   IROWC, ICOLC, NPIVM, IAINFO, LIA, IFAIL)
 DOUBLE PRECISION   A(*), DTOL(*), C(LC)
 INTEGER            ICNTXT, N, NNZ, IROW(*), ICOL(*), NOVRLP,
1                   LFILL(*), IPIVP(*), IPIVQ(*), NNZC, LC,
2                   IROWC(LC), ICOLC(LC), NPIVM(*), IAINFO(*), LIA,
3                   IFAIL
 CHARACTER*1        PSTRAT(*), MILU(*)
```

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).
m_l° – the overall number of rows (and columns) in the diagonal blocks A_k, $k = 1, 2, \ldots, n_{\mathrm{LB}}$, assigned to the calling processor (m_l° = IAINFO(4), see IAINFO).
n_{LB} – the number of row blocks assigned to the calling processor (n_{LB} = IAINFO(8), see IAINFO).
$\mathrm{NNZ_d}$ – the overall number of non-zero entries in the diagonal blocks A_k, $k = 1, 2, \ldots, n_{\mathrm{LB}}$, assigned to the calling processor ($\mathrm{NNZ_d}$ = IAINFO(9), see IAINFO).

3.2 Global and Local Arguments

Global input arguments: N, 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. This data distribution is described in more detail in Section 2.5.2 of the F11 Chapter Introduction.

F11DFFP 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

A number of Library routines can be used to generate or distribute sparse matrices in cyclic row block form:

Real sparse matrix generation: F01YAFP or F01YBFP

Real sparse matrix distribution: F01XAFP

Other Library routines employ the incomplete LU factorization of the diagonal blocks for preconditioning using the block Jacobi method:

Black Box routines: F11DHFP for the solution of real general linear systems, F11JHFP for the solution of real symmetric systems;

Preconditioning routines: F11DGFP for the solution of the preconditioning equations, generally used in conjunction with the basic solver suite comprising F11BAFP, F11BBFP and F11BCFP.

3.5 Requisites

The sparse matrix A must have been preprocessed to set up the auxiliary information vector IAINFO by F11ZBFP.

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: 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: N ≥ 1.

3: NNZ — INTEGER *Local Input*

On entry: the number of non-zero entries of 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: NNZ > 0.

4: 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. These local non-zero entries must have been reordered by a prior call to F11YAFP or F11ZBFP.

5: IROW($*$) — INTEGER array *Local Input*

6: 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 A. The contents of the arrays IROW and ICOL **must not** be changed between successive calls to library routines involving the matrix A.

7: NOVRLP — INTEGER *Global Input*

Note: a parameter indicating the amount of overlap required for additive Schwarz preconditioning. Not implemented at this release.

8: LFILL($*$) — INTEGER array *Local Input*

Note: the dimension of the array LFILL must be at least max(1,n_{LB}).

On entry: if LFILL(k) ≥ 0, for $k = 1, 2, \ldots, n_{\mathrm{LB}}$, LFILL($k$) is the maximum level of fill allowed in the decomposition of the diagonal block A_k assigned to the calling processor (see Section 6.2). A negative value of LFILL(k) indicates that DTOL(k) will be used to control the fill instead.

9: DTOL($*$) — DOUBLE PRECISION array *Local Input*

Note: the dimension of the array DTOL must be at least max(1,n_{LB}).

On entry: if LFILL(k) < 0, for $k = 1, 2, \ldots, n_{\mathrm{LB}}$, then DTOL($k$) is used as a drop tolerance to control the fill-in allowed in the decomposition of the diagonal block A_k assigned to the calling processor (see Section 6.2). Otherwise DTOL(k) is not referenced.

Constraint: DTOL(k) ≥ 0.0 if LFILL(k) < 0, for $k = 1, 2, \ldots, n_{\mathrm{LB}}$.

10: PSTRAT($*$) — CHARACTER*1 array *Local Input*

Note: the dimension of the array PSTRAT must be at least max(1,n_{LB}).

On entry: PSTRAT(k), for $k = 1, 2, \ldots, n_{\mathrm{LB}}$, specifies the pivoting strategy to be adopted for the diagonal block A_k as follows:

if PSTRAT(k) = 'N', then no pivoting is carried out;
if PSTRAT(k) = 'U', then pivoting is carried out according to the user-defined input values of IPIVP and IPIVQ;
if PSTRAT(k) = 'P', then partial pivoting by rows for stability is carried out;
if PSTRAT(k) = 'C', then 'complete' pivoting by columns for sparsity and by rows for stability is carried out.

Suggested value: PSTRAT(k) = 'C', for $k = 1, 2, \ldots, n_{\mathrm{LB}}$.

Constraint: PSTRAT(k) = 'N', 'U', 'P', or 'C', for $k = 1, 2, \ldots, n_{\mathrm{LB}}$.

11: MILU(∗) — CHARACTER*1 array *Local Input*

Note: the dimension of the array MILU must be at least max(1,n_{LB}).

On entry: MILU(k), for $k = 1, 2, \ldots, n_{\mathrm{LB}}$, indicates whether or not the factorization of A_k should be modified to preserve row-sums (see Section 6.2):

if MILU(k) = 'M', the factorization is modified;
if MILU(k) = 'N', the factorization is not modified.

Constraint: MILU(k) = 'M' or 'N', for $k = 1, 2, \ldots, n_{\mathrm{LB}}$.

12: IPIVP(∗) — INTEGER array *Local Input/Local Output*

13: IPIVQ(∗) — INTEGER array *Local Input/Local Output*

Note: the dimension of the arrays IPIVP and IPIVQ must be at least max(1,m_l^{o}).

On entry: if PSTRAT(k) = 'U', for one or more $k = 1, 2, \ldots, n_{\mathrm{LB}}$, then the corresponding parts of IPIVP and IPIVQ specify the local row and column indices of the entries to be used as pivots in the factorization of A_k.

Constraint: The input values of IPIVP and IPIVQ must have been returned from a prior call to F11DFFP involving a matrix with sparsity pattern identical to A.

On exit: the local row and column indices of the entries used as pivots.

14: NNZC — INTEGER array *Local Output*

On exit: the number of non-zero entries in the matrices C_k, $k = 1, 2, \ldots, n_{\mathrm{LB}}$, assigned to the calling processor.

15: C(LC) — DOUBLE PRECISION array *Local Output*

On exit: the first NNZC entries of C store the non-zero entries of the matrices C_k, $k = 1, 2, \ldots, n_{\mathrm{LB}}$, assigned to the calling processor.

16: LC — INTEGER *Local Input*

On entry: the dimension of the arrays C, IROWC and ICOLC as declared in the (sub)program from which F11DFFP is called.

Suggested value: depending on the arguments LFILL and DTOL, a small multiple of NNZ, LC$\leq$ 5 $\times$ NNZ, should be adequate for most problems.

Constraint: LC $\geq$ max(1,$\mathrm{NNZ_d}$).

17: IROWC(LC) — INTEGER array *Local Output*

18: ICOLC(LC) — INTEGER array *Local Output*

On exit: the first NNZC entries of IROWC and ICOLC store the local row and column indices of the non-zero entries of the matrices C_k, $k = 1, 2, \ldots, n_{\mathrm{LB}}$, returned in C.

19: NPIVM(∗) — INTEGER array *Local Output*

On exit: NPIVM(k), for $k = 1, 2, \ldots, n_{\mathrm{LB}}$, specifies the number of pivots which were modified during the factorization of A_k to ensure that the matrix M_k assigned to the calling processor exists. The quality of the local contribution to the preconditioner will generally depend on the returned value of NPIVM(k). If NPIVM(k) > 0, the local contribution to the preconditioner may not be satisfactory. In this case it may be advantageous to call F11DFFP again with an increased value of LFILL(k), a reduced value of DTOL(k), or PSTRAT(k) set to 'C'.

20: IAINFO(∗) — INTEGER array *Local Input/Local Output*

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

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.

On exit: auxiliary information about the matrix A including information needed to efficiently solve the equations $Mx = y$ and $M^T x = y$, where M is the matrix formed by the diagonal blocks M_k. 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 = 5 or 7, IAINFO(1) may be **smaller** than the minimum required value of LIA, but still gives a valuable indication of the minimum value of LIA to be used in future program runs.

21: LIA — INTEGER *Local Input*

Note: if LIA is set to −1, F11DFFP will allocate the memory required for storing the auxiliary information about the matrix A (see Section 6.4). In this case, LIA is treated as a **global** input, i.e., LIA = −1 must be supplied to all processors.

On entry: if LIA ≠ −1, the dimension of the array IAINFO as declared in the (sub)program from which F11DFFP is called.

Suggested value: LIA = −1 unless the required value of LIA was determined by a prior call to F11DFFP.

Constraint: LIA = −1 or LIA ≥ IAINFO(2). The choice between these two options is determined by the choice made at the call to F11ZBFP.

22: 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 initialised by a prior call to F11ZBFP.

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 initialised by a call to F11ZBFP, at least one of these arguments was changed before calling F11DFFP.

IFAIL $= 3$

For at least one $k = 1, 2, \ldots, n_{\mathrm{LB}}$, PSTRAT($k$) = 'U', but the corresponding part of IPIVP and/or IPIVQ does not hold a valid permutation of the index set associated with the diagonal block A_k. This indicates that IPIVP and/or IPIVQ were not obtained by or have been modified since a prior call to F11DFFP involving a matrix with sparsity pattern identical to A.

5.2 Any Error Checking Mode

IFAIL $= 4$

LC is too small, resulting in insufficient storage space for the non-zero entries in the diagonal blocks C_k.

IFAIL $= 5$

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

IFAIL $= 6$

A serious error has occurred in an internal call to an auxiliary routine. Check all subroutine calls and array sizes. Seek expert help.

IFAIL $= 7$

An error in the internal memory allocation mechanism occurred.

6 Further Comments

6.1 Accuracy

The accuracy of each factorization is to a large extent determined by the size of the elements that are dropped and by the size of any modifications made to the pivot entries. If both these are small, then the computed incomplete factors will provide a good representation of the original diagonal blocks. In general, increasing the levels of fill, the input parameter LFILL, or reducing the drop tolerances, the input parameter DTOL, can lead to better preconditioners.

If F11DFFP is used in combination with F11BBFP or F11DHFP, a better preconditioner will result in fewer iterations required for convergence. However, the cost of the factorization will also increase.

6.2 Control of Fill-in

If LFILL(k) ≥ 0 the amount of fill-in occurring in the incomplete factorization of A_k is controlled by limiting the maximum **level** of fill-in to LFILL(k). The original non-zero entries of A_k are defined to be of level zero. The fill level of a new non-zero location occurring during the factorization is defined as:

$$L = \max(L_e, L_c) + 1,$$

where L_e is the level of fill of the element being eliminated, and L_c is the level of fill of the element causing the fill-in.

If LFILL(k) < 0 the fill-in is controlled by means of the **drop tolerance** DTOL(k). A potential fill-in element $a_{ij}^{(k)}$ occurring in row i and column j will not be included if:

$$|a_{ij}^{(k)}| < \text{DTOL}(k) \times \alpha_k,$$

where α_k is the maximum absolute value of any element in the matrix A_k.

For either method of control, any entries which are not included are discarded unless MILU(k) = 'M', in which case their contributions are subtracted from the pivot element in the relevant elimination row, to preserve the row-sums of the original matrix A_k.

6.3 Choice of Parameters

In general, it is impossible to provide guidance for the optimal choice of the parameters required by this routine: these vary with the type of problem, perhaps some experimentation is required for each matrix type encountered.

If the diagonal blocks A_k are not known to have any particular special properties, the following strategy is recommended. Start with LFILL(k) = 0 and PSTRAT(k) = 'C', for $k = 1, 2, \ldots, n_{\text{LB}}$. If a value returned for NPIVM(k) is significantly larger than zero, i.e., a large number of pivot modifications were required to ensure that the local M_k existed, the preconditioner is not likely to be satisfactory. In this case increase LFILL(k) until NPIVM(k) falls to a value close to zero.

If a diagonal block A_k has non-positive off-diagonal entries, is non-singular, and has only non-negative entries in its inverse, it is called an 'M-matrix'. It can be shown that no pivot modifications are required in the incomplete LU factorization of an M-matrix (Saad [1]). In this case a good preconditioner can generally be expected by setting LFILL(k) = 0, PSTRAT(k) = 'N' and MILU(k) = 'M'.

6.4 Releasing Internally Allocated Resources

F11DFFP will internally allocate 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

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

8 Example

See Section 8 of the document for F11DHFP.