NAG Library Routine Document

F11DAF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

F11DAF computes an incomplete LU factorization of a real sparse nonsymmetric matrix, represented in coordinate storage format. This factorization may be used as a preconditioner in combination with F11BEF or F11DCF.

2 Specification

```
SUBROUTINE F11DAF (N, NNZ, A, LA, IROW, ICOL, LFILL, DTOL, PSTRAT, MILU,    &
                   IPIVP, IPIVQ, ISTR, IDIAG, NNZC, NPIVM, IWORK,           &
                   LIWORK, IFAIL)

INTEGER            N, NNZ, LA, IROW(LA), ICOL(LA), LFILL, IPIVP(N),         &
                   IPIVQ(N), ISTR(N+1), IDIAG(N), NNZC, NPIVM,              &
                   IWORK(LIWORK), LIWORK, IFAIL
REAL (KIND=nag_wp) A(LA), DTOL
CHARACTER(1)       PSTRAT, MILU
```

3 Description

F11DAF computes an incomplete LU factorization (see Meijerink and Van der Vorst (1977) and Meijerink and Van der Vorst (1981)) of a real sparse nonsymmetric n by n matrix A. The factorization is intended primarily for use as a preconditioner with one of the iterative solvers F11BEF or F11DCF.

The decomposition is written in the form

$$A = M + R$$

where

$$M = PLDUQ$$

and L is lower triangular with unit diagonal elements, D is diagonal, U is upper triangular with unit diagonals, P and Q are permutation matrices, and R is a remainder matrix.

The amount of fill-in occurring in the factorization can vary from zero to complete fill, and can be controlled by specifying either the maximum level of fill LFILL, or the drop tolerance DTOL.

The argument PSTRAT defines the pivoting strategy to be used. The options currently available are no pivoting, user-defined pivoting, partial pivoting by columns for stability, and complete pivoting by rows for sparsity and by columns for stability. The factorization may optionally be modified to preserve the row-sums of the original matrix.

The sparse matrix A is represented in coordinate storage (CS) format (see Section 2.1.1 in the F11 Chapter Introduction). The array A stores all the nonzero elements of the matrix A, while arrays IROW and ICOL store the corresponding row and column indices respectively. Multiple nonzero elements may not be specified for the same row and column index.

The preconditioning matrix M is returned in terms of the CS representation of the matrix

$$C = L + D^{-1} + U - 2I.$$

Further algorithmic details are given in Section 9.3.

4 References

Meijerink J and Van der Vorst H (1977) An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix *Math. Comput.* **31** 148–162

Meijerink J and Van der Vorst H (1981) Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems *J. Comput. Phys.* **44** 134–155

Salvini S A and Shaw G J (1996) An evaluation of new NAG Library solvers for large sparse unsymmetric linear systems *NAG Technical Report TR2/96*

5 Arguments

1: N – INTEGER *Input*

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

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

2: NNZ – INTEGER *Input*

On entry: the number of nonzero elements in the matrix A.

Constraint: $1 \leq \mathrm{NNZ} \leq \mathrm{N}^2$.

3: A(LA) – REAL (KIND=nag_wp) array *Input/Output*

On entry: the nonzero elements in the matrix A, ordered by increasing row index, and by increasing column index within each row. Multiple entries for the same row and column indices are not permitted. The routine F11ZAF may be used to order the elements in this way.

On exit: the first NNZ entries of A contain the nonzero elements of A and the next NNZC entries contain the elements of the matrix C. Matrix elements are ordered by increasing row index, and by increasing column index within each row.

4: LA – INTEGER *Input*

On entry: the dimension of the arrays A, IROW and ICOL as declared in the (sub)program from which F11DAF is called. These arrays must be of sufficient size to store both A (NNZ elements) and C (NNZC elements).

Constraint: $\mathrm{LA} \geq 2 \times \mathrm{NNZ}$.

5: IROW(LA) – INTEGER array *Input/Output*

6: ICOL(LA) – INTEGER array *Input/Output*

On entry: the row and column indices of the nonzero elements supplied in A.

Constraints:

IROW and ICOL must satisfy these constraints (which may be imposed by a call to F11ZAF):

$1 \leq \mathrm{IROW}(i) \leq \mathrm{N}$ and $1 \leq \mathrm{ICOL}(i) \leq \mathrm{N}$, for $i = 1, 2, \ldots, \mathrm{NNZ}$;
either $\mathrm{IROW}(i-1) < \mathrm{IROW}(i)$ or both $\mathrm{IROW}(i-1) = \mathrm{IROW}(i)$ and $\mathrm{ICOL}(i-1) < \mathrm{ICOL}(i)$, for $i = 2, 3, \ldots, \mathrm{NNZ}$.

On exit: the row and column indices of the nonzero elements returned in A.

7: LFILL – INTEGER *Input*

On entry: if $\mathrm{LFILL} \geq 0$ its value is the maximum level of fill allowed in the decomposition (see Section 9.2). A negative value of LFILL indicates that DTOL will be used to control the fill instead.

8: DTOL – REAL (KIND=nag_wp) *Input*

On entry: if LFILL < 0, DTOL is used as a drop tolerance to control the fill-in (see Section 9.2); otherwise DTOL is not referenced.

Constraint: if LFILL < 0, DTOL ≥ 0.0.

9: PSTRAT – CHARACTER(1) *Input*

On entry: specifies the pivoting strategy to be adopted.

PSTRAT = 'N'
No pivoting is carried out.

PSTRAT = 'U'
Pivoting is carried out according to the user-defined input values of IPIVP and IPIVQ.

PSTRAT = 'P'
Partial pivoting by columns for stability is carried out.

PSTRAT = 'C'
Complete pivoting by rows for sparsity, and by columns for stability, is carried out.

Suggested value: PSTRAT = 'C'.

Constraint: PSTRAT = 'N', 'U', 'P' or 'C'.

10: MILU – CHARACTER(1) *Input*

On entry: indicates whether or not the factorization should be modified to preserve row-sums (see Section 9.4).

MILU = 'M'
The factorization is modified.

MILU = 'N'
The factorization is not modified.

Constraint: MILU = 'M' or 'N'.

11: IPIVP(N) – INTEGER array *Input/Output*
12: IPIVQ(N) – INTEGER array *Input/Output*

On entry: if PSTRAT = 'U', then IPIVP(k) and IPIVQ(k) must specify the row and column indices of the element used as a pivot at elimination stage k. Otherwise IPIVP and IPIVQ need not be initialized.

Constraint: if PSTRAT = 'U', IPIVP and IPIVQ must both hold valid permutations of the integers on [1,N].

On exit: the pivot indices. If IPIVP(k) $= i$ and IPIVQ(k) $= j$ then the element in row i and column j was used as the pivot at elimination stage k.

13: ISTR(N + 1) – INTEGER array *Output*

On exit: ISTR(i), for $i = 1, 2, \ldots,$ N, is the starting address in the arrays A, IROW and ICOL of row i of the matrix C. ISTR(N + 1) is the address of the last nonzero element in C plus one.

14: IDIAG(N) – INTEGER array *Output*

On exit: IDIAG(i), for $i = 1, 2, \ldots,$ N, holds the index of arrays A, IROW and ICOL which holds the diagonal element in row i of the matrix C.

15: NNZC – INTEGER *Output*

On exit: the number of nonzero elements in the matrix C.

16: NPIVM – INTEGER *Output*

On exit: if NPIVM > 0 it gives the number of pivots which were modified during the factorization to ensure that M exists.

If NPIVM $= -1$ no pivot modifications were required, but a local restart occurred (see Section 9.3). The quality of the preconditioner will generally depend on the returned value of NPIVM.

If NPIVM is large the preconditioner may not be satisfactory. In this case it may be advantageous to call F11DAF again with an increased value of LFILL, a reduced value of DTOL, or set PSTRAT = 'C'. See also Section 9.5.

17: IWORK(LIWORK) – INTEGER array *Workspace*

18: LIWORK – INTEGER *Input*

On entry: the dimension of the array IWORK as declared in the (sub)program from which F11DAF is called.

Constraint: LIWORK $\geq 7 \times$ N $+ 2$.

19: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this argument, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, N < 1,
or NNZ < 1,
or NNZ $>$ N^2,
or LA $< 2 \times$ NNZ,
or LFILL < 0 and DTOL < 0.0,
or PSTRAT $\neq$ 'N', 'U', 'P' or 'C',
or MILU $\neq$ 'M' or 'N',
or LIWORK $< 7 \times$ N $+ 2$.

IFAIL $= 2$

On entry, the arrays IROW and ICOL fail to satisfy the following constraints:

$1 \leq \mathrm{IROW}(i) \leq \mathrm{N}$ and $1 \leq \mathrm{ICOL}(i) \leq \mathrm{N}$, for $i = 1, 2, \ldots, \mathrm{NNZ}$;

$\mathrm{IROW}(i-1) < \mathrm{IROW}(i)$ or $\mathrm{IROW}(i-1) = \mathrm{IROW}(i)$ and $\mathrm{ICOL}(i-1) < \mathrm{ICOL}(i)$, for $i = 2, 3, \ldots, \mathrm{NNZ}$.

Therefore a nonzero element has been supplied which does not lie within the matrix A, is out of order, or has duplicate row and column indices. Call F11ZAF to reorder and sum or remove duplicates.

IFAIL $= 3$

On entry, PSTRAT = 'U', but one or both of IPIVP and IPIVQ does not represent a valid permutation of the integers in [1,N]. An input value of IPIVP or IPIVQ is either out of range or repeated.

IFAIL $= 4$

LA is too small, resulting in insufficient storage space for fill-in elements. The decomposition has been terminated before completion. Either increase LA or reduce the amount of fill by reducing LFILL, or increasing DTOL.

IFAIL $= 5$ (F11ZAF)

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

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

The accuracy of the factorization will be determined by the size of the elements that are dropped and the size of any modifications made to the pivot elements. If these sizes are small then the computed factors will correspond to a matrix close to A. The factorization can generally be made more accurate by increasing LFILL, or by reducing DTOL with LFILL < 0.

If F11DAF is used in combination with F11BEF or F11DCF, the more accurate the factorization the fewer iterations will be required. However, the cost of the decomposition will also generally increase.

8 Parallelism and Performance

F11DAF is not threaded in any implementation.

9 Further Comments

9.1 Timing

The time taken for a call to F11DAF is roughly proportional to $(\mathrm{NNZC})^2/\mathrm{N}$.

9.2 Control of Fill-in

If LFILL ≥ 0 the amount of fill-in occurring in the incomplete factorization is controlled by limiting the maximum **level** of fill-in to LFILL. The original nonzero elements of A are defined to be of level 0. The fill level of a new nonzero location occurring during the factorization is defined as:

$$k = \max(k_e, k_c) + 1,$$

where k_e is the level of fill of the element being eliminated, and k_c is the level of fill of the element causing the fill-in.

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

$$\left|a_{ij}\right| < \text{DTOL} \times \alpha,$$

where α is the maximum absolute value element in the matrix A.

For either method of control, any elements which are not included are discarded unless MILU = '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.

Should the factorization process break down a local restart process is implemented as described in Section 9.3. This will affect the amount of fill present in the final factorization.

9.3 Algorithmic Details

The factorization is constructed row by row. At each elimination stage a row index is chosen. In the case of complete pivoting this index is chosen in order to reduce fill-in. Otherwise the rows are treated in the order given, or some user-defined order.

The chosen row is copied from the original matrix A and modified according to those previous elimination stages which affect it. During this process any fill-in elements are either dropped or kept according to the values of LFILL or DTOL. In the case of a modified factorization (MILU = 'M') the sum of the dropped terms for the given row is stored.

Finally the pivot element for the row is chosen and the multipliers are computed for this elimination stage. For partial or complete pivoting the pivot element is chosen in the interests of stability as the element of largest absolute value in the row. Otherwise the pivot element is chosen in the order given, or some user-defined order.

If the factorization breaks down because the chosen pivot element is zero, or there is no nonzero pivot available, a local restart recovery process is implemented. The modification of the given pivot row according to previous elimination stages is repeated, but this time keeping all fill. Note that in this case the final factorization will include more fill than originally specified by the user-supplied value of LFILL or DTOL. The local restart usually results in a suitable nonzero pivot arising. The original criteria for dropping fill-in elements is then resumed for the next elimination stage (hence the **local** nature of the restart process). Should this restart process also fail to produce a nonzero pivot element an arbitrary unit pivot is introduced in an arbitrarily chosen column. F11DAF returns an integer argument NPIVM which gives the number of these arbitrary unit pivots introduced. If no pivots were modified but local restarts occurred NPIVM $= -1$ is returned.

9.4 Choice of Arguments

There is unfortunately no choice of the various algorithmic arguments which is optimal for all types of matrix, and some experimentation will generally be required for each new type of matrix encountered.

If the matrix A is not known to have any particular special properties the following strategy is recommended. Start with LFILL $= 0$ and PSTRAT = 'C'. If the value returned for NPIVM is significantly larger than zero, i.e., a large number of pivot modifications were required to ensure that M existed, the preconditioner is not likely to be satisfactory. In this case increase LFILL until NPIVM falls to a value close to zero.

If A has non-positive off-diagonal elements, is nonsingular, and has only non-negative elements 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 (see Meijerink and Van der Vorst (1977)). In this case a good preconditioner can generally be expected by setting LFILL $= 0$, PSTRAT = 'N' and MILU = 'M'.

Some illustrations of the application of F11DAF to linear systems arising from the discretization of two-dimensional elliptic partial differential equations, and to random-valued randomly structured linear systems, can be found in Salvini and Shaw (1996).

9.5 Direct Solution of Sparse Linear Systems

Although it is not their primary purpose F11DAF and F11DBF may be used together to obtain a **direct** solution to a nonsingular sparse linear system. To achieve this the call to F11DBF should be preceded by a **complete** LU factorization

$$A = PLDUQ = M.$$

A complete factorization is obtained from a call to F11DAF with LFILL < 0 and DTOL $= 0.0$, provided NPIVM ≤ 0 on exit. A positive value of NPIVM indicates that A is singular, or ill-conditioned. A factorization with positive NPIVM may serve as a preconditioner, but will not result in a direct solution. It is therefore **essential** to check the output value of NPIVM if a direct solution is required.

The use of F11DAF and F11DBF as a direct method is illustrated in Section 10 in F11DBF.

10 Example

This example reads in a sparse matrix A and calls F11DAF to compute an incomplete LU factorization. It then outputs the nonzero elements of both A and $C = L + D^{-1} + U - 2I$.

The call to F11DAF has LFILL $= 0$, and PSTRAT $=$ 'C', giving an unmodified zero-fill LU factorization, with row pivoting for sparsity and column pivoting for stability.

10.1 Program Text

```
    Program f11dafe

!     F11DAF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: f11daf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: dtol
      Integer                          :: i, ifail, la, lfill, liwork, n, nnz, &
                                          nnzc, npivm
      Character (1)                    :: milu, pstrat
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:)
      Integer, Allocatable             :: icol(:), idiag(:), ipivp(:),         &
                                          ipivq(:), irow(:), istr(:), iwork(:)
!     .. Executable Statements ..
      Write (nout,*) 'F11DAF Example Program Results'
      Write (nout,*)

!     Skip heading in data file

      Read (nin,*)

!     Read algorithmic parameters

      Read (nin,*) n
      Read (nin,*) nnz

      la = 2*nnz
      liwork = 7*n + 2

      Allocate (a(la),icol(la),idiag(n),ipivp(n),ipivq(n),irow(la),istr(n+1),  &
        iwork(liwork))
      Read (nin,*) lfill, dtol
      Read (nin,*) pstrat
      Read (nin,*) milu
```

```
!     Read the matrix A

      Do i = 1, nnz
        Read (nin,*) a(i), irow(i), icol(i)
      End Do

!     Calculate incomplete LU factorization

!     ifail: behaviour on error exit
!             =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
      Call f11daf(n,nnz,a,la,irow,icol,lfill,dtol,pstrat,milu,ipivp,ipivq,     &
        istr,idiag,nnzc,npivm,iwork,liwork,ifail)

!     Output original matrix

      Write (nout,*) ' Original Matrix'
      Write (nout,'(A,I4)') ' N     =', n
      Write (nout,'(A,I4)') ' NNZ   =', nnz
      Do i = 1, nnz
        Write (nout,'(I8,E16.4,2I8)') i, a(i), irow(i), icol(i)
      End Do
      Write (nout,*)

!     Output details of the factorization

      Write (nout,*) ' Factorization'
      Write (nout,'(A,I4)') ' N     =', n
      Write (nout,'(A,I4)') ' NNZ   =', nnzc
      Write (nout,'(A,I4)') ' NPIVM =', npivm
      Do i = nnz + 1, nnz + nnzc
        Write (nout,'(I8,E16.4,2I8)') i, a(i), irow(i), icol(i)
      End Do
      Write (nout,*)

      Write (nout,*) '        I       IPIVP(I)  IPIVQ(I)'
      Do i = 1, n
        Write (nout,'(3I10)') i, ipivp(i), ipivq(i)
      End Do

    End Program f11dafe
```

10.2 Program Data

```
F11DAF Example Program Data
  4                   N
  11                  NNZ
  0 0.0               LFILL, DTOL
  'C'                 PSTRAT
  'N'                 MILU
  1.   1    2
  1.   1    3
 -1.   2    1
  2.   2    3
  2.   2    4
  3.   3    1
 -2.   3    4
  1.   4    1
 -2.   4    2
  1.   4    3
  1.   4    4        A(I), IROW(I), ICOL(I), I=1,...,NNZ
```

10.3 Program Results

```
 F11DAF Example Program Results

  Original Matrix
 N     =   4
 NNZ   =  11
       1      0.1000E+01       1       2
       2      0.1000E+01       1       3
       3     -0.1000E+01       2       1
       4      0.2000E+01       2       3
       5      0.2000E+01       2       4
       6      0.3000E+01       3       1
       7     -0.2000E+01       3       4
       8      0.1000E+01       4       1
       9     -0.2000E+01       4       2
      10      0.1000E+01       4       3
      11      0.1000E+01       4       4

  Factorization
 N     =   4
 NNZ   =  11
 NPIVM =   0
      12      0.1000E+01       1       1
      13      0.1000E+01       1       3
      14      0.3333E+00       2       2
      15     -0.6667E+00       2       4
      16     -0.3333E+00       3       2
      17      0.5000E+00       3       3
      18      0.6667E+00       3       4
      19     -0.2000E+01       4       1
      20      0.3333E+00       4       2
      21      0.1500E+01       4       3
      22     -0.3000E+01       4       4

         I       IPIVP(I)  IPIVQ(I)
         1         1         2
         2         3         1
         3         2         3
         4         4         4
```