# NAG Fortran Library Routine Document

# F11DNF

**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

F11DNF computes an incomplete $LU$ factorization of a complex sparse non-Hermitian matrix, represented in coordinate storage format. This factorization may be used as a preconditioner in combination with F11DQF or F11BSF.

## 2 Specification

```
 SUBROUTINE F11DNF(N, NNZ, A, LA, IROW, ICOL, LFILL, DTOL, PSTRAT, MILU,
1                  IPIVP, IPIVQ, ISTR, IDIAG, NNZC, NPIVM, IWORK, LIWORK,
2                  IFAIL)
 INTEGER           N, NNZ, LA, IROW(LA), ICOL(LA), LFILL, IPIVP(N),
1                  IPIVQ(N), ISTR(N+1), IDIAG(N), NNZC, NPIVM,
2                  IWORK(LIWORK), LIWORK, IFAIL
 real              DTOL
 complex           A(LA)
 CHARACTER*1       PSTRAT, MILU
```

## 3 Description

This routine computes an incomplete $LU$ factorization (Meijerink and Van der Vorst (1977), Meijerink and Van der Vorst (1981)) of a complex sparse non-Hermitian $n$ by $n$ matrix $A$. The factorization is intended primarily for use as a preconditioner with one of the iterative solvers F11DQF or F11BSF.

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 of the F11 Chapter Introduction). The array A stores all the non-zero elements of the matrix $A$, while arrays IROW and ICOL store the corresponding row and column indices respectively. Multiple non-zero 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 8.3.

## 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

## 5 Parameters

1: N – INTEGER *Input*

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

*Constraint*: N $\geq 1$.

2: NNZ – INTEGER *Input*

*On entry*: the number of non-zero elements in the matrix $A$.

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

3: A(LA) – ***complex*** array *Input/Output*

*On entry*: the non-zero 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 F11ZNF may be used to order the elements in this way.

*On exit*: the first NNZ entries of A contain the non-zero 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 F11DNF is called. These arrays must be of sufficient size to store both $A$ (NNZ elements) and $C$ (NNZC elements).

*Constraint*: LA $\geq 2 \times$ NNZ.

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

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

*On entry*: the row and column indices of the non-zero elements supplied in A.

*Constraints*: IROW and ICOL must satisfy the following constraints (which may be imposed by a call to F11ZNF):

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

*On exit*: the row and column indices of the non-zero elements returned in A.

7: LFILL – INTEGER *Input*

*On entry*: if LFILL $\geq 0$ its value is the maximum level of fill allowed in the decomposition (see Section 8.2). A negative value of LFILL indicates that DTOL will be used to control the fill instead.

8: DTOL – ***real*** *Input*

*On entry*: if LFILL $< 0$ then DTOL is used as a drop tolerance to control the fill-in (see Section 8.2). Otherwise DTOL is not referenced.

*Constraint*: DTOL $\geq 0.0$ if LFILL $< 0$.

9: PSTRAT – CHARACTER*1 *Input*

*On entry*: specifies the pivoting strategy to be adopted as follows:

if PSTRAT = 'N', then no pivoting is carried out;

if PSTRAT = 'U', then pivoting is carried out according to the user-defined input values of IPIVP and IPIVQ;

if PSTRAT = 'P', then partial pivoting by columns for stability is carried out.

if PSTRAT = 'C', then 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 8.4):

if MILU = 'M', the factorization is modified (MILU);

if 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 initialised.

*Constraint*: if PSTRAT = 'U', then 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 holds the index of arrays A, IROW and ICOL where row $i$ of the matrix $C$ starts. ISTR(N + 1) holds the address of the last non-zero 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 non-zero 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 8.3). 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 F11DNF again with an increased value of LFILL, a reduced value of DTOL, or PSTRAT set to 'C'. See also Section 8.5.

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 F11DNF is called.

*Constraint*: LIWORK $\geq 7 \times \mathrm{N} + 2$.

19: IFAIL – INTEGER *Input/Output*

*On entry*: IFAIL must be set to $0$, $-1$ or $1$. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

*On exit*: IFAIL $= 0$ unless the routine detects an error (see Section 6).

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, for users not familiar with this parameter the recommended value is $0$. **When the value $-1$ or $1$ is used it is essential to test the value of IFAIL on exit.**

## 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 $> \mathrm{N}^2$,
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 $< 7 \times \mathrm{N} + 2$.

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 non-zero element has been supplied which does not lie within the matrix $A$, is out of order, or has duplicate row and column indices. Call F11ZNF 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

A serious error has occurred in an internal call to F11ZNF. Check all subroutine calls and array sizes. Seek expert help.

# 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 F11DNF is used in combination with F11BSF, or F11DQF, the more accurate the factorization the fewer iterations will be required. However, the cost of the decomposition will also generally increase.

# 8 Further Comments

## 8.1 Timing

The time taken for a call to F11DNF is roughly proportional to $(\text{NNZC})^2/\text{N}$.

## 8.2 Control of Fill-in

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

$$k = \max(k_{\text{e}}, k_{\text{c}}) + 1,$$

where $k_{\text{e}}$ is the level of fill of the element being eliminated, and $k_{\text{c}}$ 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:

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

where $\alpha$ is the maximum modulus 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 8.3. This will affect the amount of fill present in the final factorization.

## 8.3 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 non-zero 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 non-zero 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 non-zero pivot element an arbitrary unit pivot is introduced in an arbitrarily chosen column. F11DNF returns an integer parameter NPIVM which gives the number of these arbitrary unit pivots introduced. If no pivots were modified but local restarts occurred NPIVM $= -1$ is returned.

### 8.4 Choice of Parameters

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

For certain classes of matrices (typically those arising from the discretisation of elliptic or parabolic partial differential equations) the convergence rate of the preconditioned iterative solver can sometimes be significantly improved by using an incomplete factorization which preserves the row-sums of the original matrix. In these cases try setting MILU $=$ 'M'.

### 8.5 Direct Solution of Sparse Linear Systems

Although it is not their primary purpose F11DNF and F11DPF may be used together to obtain a **direct** solution to a non-singular sparse complex non-Hermitian linear system. To achieve this the call to F11DPF should be preceded by a **complete** $LU$ factorization

$$A = PLDUQ = M.$$

A complete factorization is obtained from a call to F11DNF with LFILL $< 0$ and DTOL $= 0.0$, provided NPIVM $\leq 0$ 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 F11DNF and F11DPF as a direct method is illustrated in F11DPF.

## 9 Example

This example program reads in a complex sparse non-Hermitian matrix $A$ and calls F11DNF to compute an incomplete $LU$ factorization. It then outputs the non-zero elements of both $A$ and $C = L + D^{-1} + U - 2I$.

The call to F11DNF has LFILL $= 0$, and PSTRAT $=$ 'C', giving an unmodified zero-fill $LU$ factorization, with row pivoting for sparsity and column pivoting for stability.

### 9.1 Program Text

**Note:** the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F11DNF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LA, LIWORK
      PARAMETER        (NMAX=1000,LA=10000,LIWORK=7*NMAX+2)
*     .. Local Scalars ..
      real             DTOL
      INTEGER          I, IFAIL, LFILL, N, NNZ, NNZC, NPIVM
      CHARACTER        MILU, PSTRAT
```

```
*     .. Local Arrays ..
      complex           A(LA)
      INTEGER           ICOL(LA), IDIAG(NMAX), IPIVP(NMAX), IPIVQ(NMAX),
     +                  IROW(LA), ISTR(NMAX+1), IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL          F11DNF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11DNF Example Program Results'
      WRITE (NOUT,*)
*     Skip heading in data file
      READ (NIN,*)
*
*     Read algorithmic parameters
*
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
         READ (NIN,*) NNZ
         READ (NIN,*) LFILL, DTOL
         READ (NIN,*) PSTRAT
         READ (NIN,*) MILU
*
*        Read the matrix A
*
         DO 20 I = 1, NNZ
            READ (NIN,*) A(I), IROW(I), ICOL(I)
   20    CONTINUE
*
*        Calculate incomplete LU factorization
*
         IFAIL = 0
         CALL F11DNF(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 40 I = 1, NNZ
            WRITE (NOUT,'(I8,5X,''('',D16.4,'','',D16.4,'')'',2I8)') I,
     +        A(I), IROW(I), ICOL(I)
   40    CONTINUE
         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 60 I = NNZ + 1, NNZ + NNZC
            WRITE (NOUT,'(I8,5X,''('',D16.4,'','',D16.4,'')'',2I8)') I,
     +        A(I), IROW(I), ICOL(I)
   60    CONTINUE
         WRITE (NOUT,*)
*
         WRITE (NOUT,*) '        I       IPIVP(I)  IPIVQ(I)'
         DO 80 I = 1, N
            WRITE (NOUT,'(3I10)') I, IPIVP(I), IPIVQ(I)
   80    CONTINUE
*
      END IF
      STOP
      END
```

## 9.2 Program Data

```
F11DNF Example Program Data
  4                   N
  11                  NNZ
  0 0.0               LFILL, DTOL
  'C'                 PSTRAT
  'N'                 MILU
 ( 1., 3.)   1    2
 ( 1., 0.)   1    3
 (-1.,-2.)   2    1
 ( 2.,-2.)   2    3
 ( 2., 1.)   2    4
 ( 0., 5.)   3    1
 (-2., 0.)   3    4
 ( 1., 1.)   4    1
 (-2., 4.)   4    2
 ( 1.,-3.)   4    3
 ( 0., 7.)   4    4   A(I), IROW(I), ICOL(I), I=1,...,NNZ
```

## 9.3 Program Results

```
 F11DNF Example Program Results

  Original Matrix
 N     =   4
 NNZ   =  11
       1     (      0.1000E+01,      0.3000E+01)       1       2
       2     (      0.1000E+01,      0.0000E+00)       1       3
       3     (     -0.1000E+01,     -0.2000E+01)       2       1
       4     (      0.2000E+01,     -0.2000E+01)       2       3
       5     (      0.2000E+01,      0.1000E+01)       2       4
       6     (      0.0000E+00,      0.5000E+01)       3       1
       7     (     -0.2000E+01,      0.0000E+00)       3       4
       8     (      0.1000E+01,      0.1000E+01)       4       1
       9     (     -0.2000E+01,      0.4000E+01)       4       2
      10     (      0.1000E+01,     -0.3000E+01)       4       3
      11     (      0.0000E+00,      0.7000E+01)       4       4

  Factorization
 N     =   4
 NNZ   =  11
 NPIVM =   0
      12     (      0.1000E+00,     -0.3000E+00)       1       1
      13     (      0.1000E+00,     -0.3000E+00)       1       3
      14     (      0.0000E+00,     -0.2000E+00)       2       2
      15     (      0.0000E+00,      0.4000E+00)       2       4
      16     (     -0.4000E+00,      0.2000E+00)       3       2
      17     (      0.2500E+00,      0.2500E+00)       3       3
      18     (     -0.5000E-01,      0.6500E+00)       3       4
      19     (      0.1000E+01,      0.1000E+01)       4       1
      20     (      0.2000E+00,     -0.2000E+00)       4       2
      21     (      0.1000E+01,     -0.1000E+01)       4       3
      22     (     -0.4803E-01,     -0.1397E+00)       4       4

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