NAG Library Routine Document

F11DPF

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

F11DPF solves a system of complex linear equations involving the incomplete LU preconditioning matrix generated by F11DNF.

2 Specification

```
SUBROUTINE F11DPF (TRANS, N, A, LA, IROW, ICOL, IPIVP, IPIVQ, ISTR,        &
                    IDIAG, CHECK, Y, X, IFAIL)
INTEGER                 N, LA, IROW(LA), ICOL(LA), IPIVP(N), IPIVQ(N),     &
                        ISTR(N+1), IDIAG(N), IFAIL
COMPLEX (KIND=nag_wp) A(LA), Y(N), X(N)
CHARACTER(1)            TRANS, CHECK
```

3 Description

F11DPF solves a system of complex linear equations

$$Mx = y, \qquad \text{or} \qquad M^{\mathrm{T}}x = y,$$

according to the value of the argument TRANS, where the matrix $M = PLDUQ$ corresponds to an incomplete LU decomposition of a complex sparse matrix stored in coordinate storage (CS) format (see Section 2.1.1 in the F11 Chapter Introduction), as generated by F11DNF.

In the above decomposition L is a lower triangular sparse matrix with unit diagonal elements, D is a diagonal matrix, U is an upper triangular sparse matrix with unit diagonal elements and, P and Q are permutation matrices. L, D and U are supplied to F11DPF through the matrix

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

which is an N by N sparse matrix, stored in CS format, as returned by F11DNF. The permutation matrices P and Q are returned from F11DNF via the arrays IPIVP and IPIVQ.

It is envisaged that a common use of F11DPF will be to carry out the preconditioning step required in the application of F11BSF to sparse complex linear systems. F11DPF is used for this purpose by the Black Box routine F11DQF.

F11DPF may also be used in combination with F11DNF to solve a sparse system of complex linear equations directly (see Section 9.5 in F11DNF). This use of F11DPF is illustrated in Section 10.

4 References

None.

5 Arguments

1: TRANS – CHARACTER(1) *Input*

On entry: specifies whether or not the matrix M is transposed.

TRANS = 'N'
  $Mx = y$ is solved.

TRANS = 'T'
: $M^{\mathrm{T}}x = y$ is solved.

Constraint: TRANS = 'N' or 'T'.

2: N – INTEGER *Input*

On entry: n, the order of the matrix M. This **must** be the same value as was supplied in the preceding call to F11DNF.

Constraint: N $\geq$ 1.

3: A(LA) – COMPLEX (KIND=nag_wp) array *Input*

On entry: the values returned in the array A by a previous call to F11DNF.

4: LA – INTEGER *Input*

On entry: the dimension of the arrays A, IROW and ICOL as declared in the (sub)program from which F11DPF is called. This **must** be the same value supplied in the preceding call to F11DNF.

5: IROW(LA) – INTEGER array *Input*
6: ICOL(LA) – INTEGER array *Input*
7: IPIVP(N) – INTEGER array *Input*
8: IPIVQ(N) – INTEGER array *Input*
9: ISTR(N + 1) – INTEGER array *Input*
10: IDIAG(N) – INTEGER array *Input*

On entry: the values returned in arrays IROW, ICOL, IPIVP, IPIVQ, ISTR and IDIAG by a previous call to F11DNF.

11: CHECK – CHARACTER(1) *Input*

On entry: specifies whether or not the CS representation of the matrix M should be checked.

CHECK = 'C'
: Checks are carried on the values of N, IROW, ICOL, IPIVP, IPIVQ, ISTR and IDIAG.

CHECK = 'N'
: None of these checks are carried out.

See also Section 9.2.

Constraint: CHECK = 'C' or 'N'.

12: Y(N) – COMPLEX (KIND=nag_wp) array *Input*

On entry: the right-hand side vector y.

13: X(N) – COMPLEX (KIND=nag_wp) array *Output*

On exit: the solution vector x.

14: 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, TRANS $\neq$ 'N' or 'T',
or CHECK $\neq$ 'C' or 'N'.

IFAIL $= 2$

On entry, N < 1.

IFAIL $= 3$

On entry, the CS representation of the preconditioning matrix M is invalid. Further details are given in the error message. Check that the call to F11DPF has been preceded by a valid call to F11DNF and that the arrays A, IROW, ICOL, IPIVP, IPIVQ, ISTR and IDIAG have not been corrupted between the two calls.

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

If TRANS = 'N' the computed solution x is the exact solution of a perturbed system of equations $(M + \delta M)x = y$, where

$$|\delta M| \leq c(n)\epsilon P|L||D||U|Q,$$

$c(n)$ is a modest linear function of n, and ϵ is the ***machine precision***. An equivalent result holds when TRANS = 'T'.

8 Parallelism and Performance

F11DPF is not threaded in any implementation.

9 Further Comments

9.1 Timing

The time taken for a call to F11DPF is proportional to the value of NNZC returned from F11DNF.

9.2 Use of CHECK

It is expected that a common use of F11DPF will be to carry out the preconditioning step required in the application of F11BSF to sparse complex linear systems. In this situation F11DPF is likely to be called many times with the same matrix M. In the interests of both reliability and efficiency, you are recommended to set CHECK = 'C' for the first of such calls, and to set CHECK = 'N' for all subsequent calls.

10 Example

This example reads in a complex sparse non-Hermitian matrix A and a vector y. It then calls F11DNF, with LFILL $= -1$ and DTOL $= 0.0$, to compute the **complete** LU decomposition

$$A = PLDUQ.$$

Finally it calls F11DPF to solve the system

$$PLDUQx = y.$$

10.1 Program Text

```
    Program f11dpfe

!     F11DPF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: f11dnf, f11dpf, 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)                     :: check, milu, pstrat, trans
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:), x(:), y(:)
      Integer, Allocatable              :: icol(:), idiag(:), ipivp(:),        &
                                           ipivq(:), irow(:), istr(:), iwork(:)
!     .. Executable Statements ..
      Write (nout,*) 'F11DPF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)

!     Read order of matrix and number of nonzero entries

      Read (nin,*) n
      Read (nin,*) nnz
      la = 3*nnz
      liwork = 7*n + 2
      Allocate (a(la),x(n),y(n),icol(la),idiag(n),ipivp(n),ipivq(n),irow(la),  &
        istr(n+1),iwork(liwork))

!     Read the matrix A

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

!     Read the vector y

      Read (nin,*) y(1:n)

!     Calculate LU factorization
```

```
      lfill = -1
      dtol = 0.0E0_nag_wp
      pstrat = 'C'
      milu = 'N'

!     ifail: behaviour on error exit
!             =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
      Call f11dnf(n,nnz,a,la,irow,icol,lfill,dtol,pstrat,milu,ipivp,ipivq,     &
        istr,idiag,nnzc,npivm,iwork,liwork,ifail)

!     Check value of NPIVM

      If (npivm>0) Then

        Write (nout,*) 'Factorization is not complete'

      Else

!       Solve P L D U x = y

        trans = 'N'
        check = 'C'

        ifail = 0
        Call f11dpf(trans,n,a,la,irow,icol,ipivp,ipivq,istr,idiag,check,y,x,   &
          ifail)

!       Output results

        Write (nout,*) 'Solution of linear system'
        Write (nout,99999) x(1:n)
      End If

99999 Format (1X,'(',E16.4,',',E16.4,')')
    End Program f11dpfe
```

10.2 Program Data

```
F11DPF Example Program Data
  4                     N
 11                     NNZ
 ( 1., 2.)   1    2
 ( 1., 3.)   1    3
 (-1.,-3.)   2    1
 ( 2., 0.)   2    3
 ( 0., 4.)   2    4
 ( 3., 4.)   3    1
 (-2., 0.)   3    4
 ( 1.,-1.)   4    1
 (-2.,-1.)   4    2
 ( 1., 0.)   4    3
 ( 1., 3.)   4    4     A(I), IROW(I), ICOL(I), I=1,...,NNZ
 (  5.0, 14.0)
 ( 21.0,  5.0)
 (-21.0, 18.0)
 ( 14.0,  4.0)          Y(I), I=1,...,N
```

10.3 Program Results

```
 F11DPF Example Program Results

 Solution of linear system
 (      0.1000E+01,      0.4000E+01)
 (      0.2000E+01,      0.3000E+01)
 (      0.3000E+01,     -0.2000E+01)
 (      0.4000E+01,     -0.1000E+01)
```