NAG Fortran Library Routine Document

F11DXF

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

F11DXF computes the **approximate** solution of a complex, Hermitian or non-Hermitian, sparse system of linear equations applying a number of Jacobi iterations. It is expected that F11DXF will be used as a preconditioner for the iterative solution of complex sparse systems of equations.

2 Specification

```
SUBROUTINE F11DXF(STORE, TRANS, INIT, NITER, N, NNZ, A, IROW, ICOL,
1                 CHECK, B, X, DIAG, WORK, IFAIL)
INTEGER           NITER, N, NNZ, IROW(NNZ), ICOL(NNZ), IFAIL
complex           A(NNZ), B(N), X(N), DIAG(N), WORK(N)
CHARACTER*1       STORE, TRANS, INIT, CHECK
```

3 Description

F11DXF computes the **approximate** solution of the complex sparse system of linear equations $Ax = b$ using NITER iterations of the Jacobi algorithm (see also Golub and van Loan (1996), Young (1971)):

$$x_{k+1} = x_k + D^{-1}(b - Ax_k) \tag{1}$$

where $k = 1, \ldots,$ NITER and $x_0 = 0$.

F11DXF can be used both for non-Hermitian and Hermitian systems of equations. For Hermitian matrices, either all non-zero elements of the matrix A can be supplied using coordinate storage (CS), or only the non-zero elements of the lower triangle of A, using symmetric coordinate storage (SCS) (see the F11 Chapter Introduction).

It is expected that F11DXF will be used as a preconditioner for the iterative solution of complex sparse systems of equations, using either the suite comprising the routines F11GRF, F11GSF and F11GTF, for Hermitian systems, or the suite comprising the routines F11BRF, F11BSF and F11BTF, for non-Hermitian systems of equations.

4 References

Golub G H and van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

Young D (1971) *Iterative Solution of Large Linear Systems* Academic Press, New York

5 Parameters

1: STORE – CHARACTER*1 *Input*

On entry: specifies whether the matrix A is stored using symmetric coordinate storage (SCS) (applicable only to a Hermitian matrix A) or coordinate storage (CS) (applicable to both Hermitian and non-Hermitian matrices).

If STORE = 'N', then the complete matrix A is stored in CS format.

If STORE = 'S', then the lower triangle of the Hermitian matrix A is stored in SCS format.

Constraint: STORE = 'N' or 'S'.

2: TRANS – CHARACTER*1 *Input*

On entry: if STORE = 'N', specifies whether the approximate solution of $Ax = b$ or of $A^T x = b$ is required:

if TRANS = 'N', then the approximate solution of $Ax = b$ is calculated;

if TRANS = 'T', then the approximate solution of $A^T x = b$ is calculated.

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

Suggested value: if the matrix A is Hermitian and stored in CS format, it is recommended that TRANS = 'N' for reasons of efficiency.

3: INIT – CHARACTER*1 *Input*

On entry: on first entry, INIT should be set to 'I', unless the diagonal elements of A are already stored in the array DIAG. Otherwise, if DIAG already contains the diagonal of A, it can be set to 'N'.

If INIT = 'N', then DIAG must contain the diagonal of A.

If INIT = 'I', then DIAG will store the diagonal of A on exit.

Constraint: INIT = 'N' or 'I'.

Suggested value: INIT = 'I' on first entry; INIT = 'N', subsequently, unless DIAG has been overwritten.

4: NITER – INTEGER *Input*

On entry: the number of Jacobi iterations requested.

Constraint: NITER > 0.

5: N – INTEGER *Input*

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

Constraint: N ≥ 1.

6: NNZ – INTEGER *Input*

On entry: if STORE = 'N', the number of non-zero elements in the matrix A. If STORE = 'S', the number of non-zero elements in the lower triangle of the matrix A.

Constraints:

if STORE = 'N', $1 \leq \text{NNZ} \leq \text{N}^2$,
If STORE = 'S', $1 \leq \text{NNZ} \leq [\text{N}(\text{N}+1)]/2$.

7: A(NNZ) – ***complex*** array *Input*

On entry: if STORE = 'N', the non-zero elements in the matrix A (CS format). If STORE = 'S', the non-zero elements in the lower triangle of the matrix A (SCS format). In both cases, the elements of either A or of its lower triangle must be ordered by increasing row index and by increasing column index within each row. Multiple entries for the same row and columns indices are not permitted. The routine F11ZNF or F11ZPF may be used to reorder the elements in this way for CS and SCS storage, respectively.

8: IROW(NNZ) – INTEGER array *Input*
9: ICOL(NNZ) – INTEGER array *Input*

On entry: if STORE = 'N', the row and column indices of the non-zero elements supplied in A. If STORE = 'S', the row and column indices of the non-zero elements of the lower triangle of the matrix A supplied in A.

Constraints:

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

10: CHECK – CHARACTER*1 *Input*

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

if CHECK = 'C', checks are carried out on the values of N, NNZ, IROW, ICOL; if INIT = 'N', DIAG is also checked;

if CHECK = 'N', none of these checks are carried out.

See also Section 8.2.

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

11: B(N) – ***complex*** array *Input*

On entry: the right-hand side vector b.

12: X(N) – ***complex*** array *Output*

On exit: the approximate solution vector x_{NITER}.

13: DIAG(N) – ***complex*** array *Input/Output*

On entry: if INIT = 'N', the diagonal elements of A.

On exit: if INIT = 'N', unchanged on exit. If INIT = 'I', the diagonal elements of A.

14: WORK(N) – ***complex*** array *Workspace*

15: 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, STORE $\neq$ 'N' or 'S',
or TRANS $\neq$ 'N' or 'T',
or INIT $\neq$ 'N' or 'I',
or CHECK $\neq$ 'C' or 'N',
or NITER ≤ 0.

IFAIL $= 2$

On entry, N < 1,
or NNZ < 1,
or NNZ $>$ N^2, if STORE $=$ 'N',
or $1 \leq$ NNZ $\leq$ [N(N $+ 1$)]/2, if STORE $=$ 'S'.

IFAIL $= 3$

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

$1 \leq$ IROW(i) $\leq$ N and

if STORE $=$ 'N' then $1 \leq$ ICOL(i) $\leq$ N, or

if STORE $=$ 'S' then $1 \leq$ ICOL(i) $\leq$ IROW(i), for $i = 1, 2, \ldots,$ NNZ.

IROW($i-1$) $<$ IROW(i) or IROW($i-1$) $=$ IROW(i) and ICOL($i-1$) $<$ ICOL(i), for $i = 2, 3, \ldots,$ 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 either F11ZAF or F11ZBF to reorder and sum or remove duplicates when STORE $=$ 'N' or STORE $=$ 'S', respectively.

IFAIL $= 4$

On entry, INIT $=$ 'N' and some diagonal elements of A stored in DIAG are zero.

IFAIL $= 5$

On entry, INIT $=$ 'I' and some diagonal elements of A are zero.

7 Accuracy

In general, the Jacobi method cannot be used on its own to solve systems of linear equations. The rate of convergence is bound by its spectral properties (see, for example, Golub and van Loan (1996)) and as a solver, the Jacobi method can only be applied to a limited set of matrices. One condition that guarantees convergence is strict diagonal dominance.

However, the Jacobi method can be used successfully as a preconditioner to a wider class of systems of equations. The Jacobi method has good vector/parallel properties, hence it can be applied very efficiently. Unfortunately, it is not possible to provide criteria which define the applicability of the Jacobi method as a preconditioner, and its usefulness must be judged for each case.

8 Further Comments

8.1 Timing

The time taken for a call to F11DXF is proportional to NITER $\times$ NNZ.

8.2 Use of CHECK

It is expected that a common use of F11DXF will be as preconditioner for the iterative solution of complex, Hermitian or non-Hermitian, linear systems. In this situation, F11DXF is likely to be called many times. In the interests of both reliability and efficiency, you are recommended to set CHECK to 'C' for the first of such calls, and to 'N' for all subsequent calls.

9 Example

This example solves the complex sparse non-Hermitian system of equations $Ax = b$ iteratively using F11DXF as a preconditioner.

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.

```
*     F11DXF Example Program Text.
*     Mark 20 Release. NAG Copyright 2001.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LA, LWORK
      PARAMETER        (NMAX=1000,LA=10000,LWORK=10*NMAX)
*     .. Local Scalars ..
      real             ANORM, SIGMAX, STPLHS, STPRHS, TOL
      INTEGER          I, IFAIL, IFAIL1, IFAILX, IREVCM, ITERM, ITN,
     +                 LWREQ, M, MAXITN, MONIT, N, NITER, NNZ
      LOGICAL          LOOP
      CHARACTER        INIT, NORM, PRECON, WEIGHT
      CHARACTER*8      METHOD
*     .. Local Arrays ..
      complex          A(LA), B(NMAX), DIAG(NMAX), WORK(LWORK), X(NMAX)
      real             WGT(NMAX)
      INTEGER          ICOL(LA), IROW(LA)
*     .. External Subroutines ..
      EXTERNAL         F11BRF, F11BSF, F11BTF, F11DXF, F11XNF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11DXF Example Program Results'
*
*     Skip heading in data file
*
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read or initialize the parameters for the iterative solver
*
         READ (NIN,*) METHOD
         READ (NIN,*) PRECON, NORM, WEIGHT, ITERM
         READ (NIN,*) M, TOL, MAXITN
         READ (NIN,*) MONIT
         ANORM = 0.0e0
         SIGMAX = 0.0e0
*
*        Read the parameters for the preconditioner
*
         READ (NIN,*) NITER
*
*        Read the number of non-zero elements of the matrix A, then read
*        the non-zero elements
*
         READ (NIN,*) NNZ
         DO 20 I = 1, NNZ
            READ (NIN,*) A(I), IROW(I), ICOL(I)
   20    CONTINUE
*
*        Read right-hand side vector b and initial approximate solution
*
         READ (NIN,*) (B(I),I=1,N)
         READ (NIN,*) (X(I),I=1,N)
*
*        Call F11BDF to initialize the solver
*
         IFAIL = 0
         CALL F11BRF(METHOD,PRECON,NORM,WEIGHT,ITERM,N,M,TOL,MAXITN,
     +               ANORM,SIGMAX,MONIT,LWREQ,WORK,LWORK,IFAIL)
*
*        Call repeatedly F11BSF to solve the equations
*        Note that the arrays B and X are overwritten
*
*        On final exit, X will contain the solution and B the residual
```

```
*        vector
*
         IFAIL = 0
         IREVCM = 0
         LOOP = .TRUE.
         INIT = 'I'
*
   40    CONTINUE
         CALL F11BSF(IREVCM,X,B,WGT,WORK,LWREQ,IFAIL)
         IF (IREVCM.EQ.-1) THEN
            IFAILX = -1
            CALL F11XNF('Transpose',N,NNZ,A,IROW,ICOL,'No checking',X,B,
     +                  IFAILX)
         ELSE IF (IREVCM.EQ.1) THEN
            IFAILX = -1
            CALL F11XNF('No transpose',N,NNZ,A,IROW,ICOL,'No checking',
     +                  X,B,IFAILX)
         ELSE IF (IREVCM.EQ.2) THEN
            IFAIL1 = -1
            CALL F11DXF('Non Hermitian','N',INIT,NITER,N,NNZ,A,IROW,
     +                  ICOL,'Check',X,B,DIAG,WORK(LWREQ+1),IFAIL1)
            INIT = 'N'
            IF (IFAIL1.NE.0) IREVCM = 6
         ELSE IF (IREVCM.EQ.3) THEN
            IFAIL1 = 0
            CALL F11BTF(ITN,STPLHS,STPRHS,ANORM,SIGMAX,WORK,LWREQ,
     +                  IFAIL1)
            WRITE (NOUT,99999) ITN, STPLHS
         ELSE IF (IREVCM.EQ.4) THEN
            LOOP = .FALSE.
         END IF
         IF (LOOP) GO TO 40
*
*        Obtain information about the computation
*
         IFAIL1 = 0
         CALL F11BTF(ITN,STPLHS,STPRHS,ANORM,SIGMAX,WORK,LWREQ,IFAIL1)
*
*        Print the output data
*
         WRITE (NOUT,99998)
         WRITE (NOUT,99997) 'Number of iterations for convergence:    ',
     +     ITN
         WRITE (NOUT,99996) 'Residual norm:                           ',
     +     STPLHS
         WRITE (NOUT,99996) 'Right-hand side of termination criterion:',
     +     STPRHS
         WRITE (NOUT,99996) '1-norm of matrix A:                      ',
     +     ANORM
*
*        Output x
*
         WRITE (NOUT,99995)
         WRITE (NOUT,99993) (X(I),I=1,N)
         WRITE (NOUT,99994)
         WRITE (NOUT,99993) (B(I),I=1,N)
      END IF
      STOP
*
99999 FORMAT (/1X,'Monitoring at iteration no.',I4,/1X,1P,'residual no',
     +       'rm: ',e14.4)
99998 FORMAT (/1X,'Final Results')
99997 FORMAT (1X,A,I4)
99996 FORMAT (1X,A,1P,e14.4)
99995 FORMAT (/2X,'  Solution vector')
99994 FORMAT (/2X,'  Residual vector')
99993 FORMAT (1X,'(',1P,e16.4,',',1P,e16.4,')')
      END
```

9.2 Program Data

```
F11DXF Example Program Data
  8                     N
  'TFQMR'               METHOD
  'P'  '1'  'N'  1      PRECON, NORM, WEIGHT, ITERM
  2  1.0e-6   20        M, TOL, MAXITN
  1                     MONIT
  2                     NITER
 24                     NNZ
 ( 2., 1.)   1    1
 (-1., 1.)   1    4
 ( 1.,-3.)   1    8
 ( 4., 7.)   2    1
 (-3., 0.)   2    2
 ( 2., 4.)   2    5
 (-7.,-5.)   3    3
 ( 2., 1.)   3    6
 ( 3., 2.)   4    1
 (-4., 2.)   4    3
 ( 0., 1.)   4    4
 ( 5.,-3.)   4    7
 (-1., 2.)   5    2
 ( 8., 6.)   5    5
 (-3.,-4.)   5    7
 (-6.,-2.)   6    1
 ( 5.,-2.)   6    3
 ( 2., 0.)   6    6
 ( 0.,-5.)   7    3
 (-1., 5.)   7    5
 ( 6., 2.)   7    7
 (-1., 4.)   8    2
 ( 2., 0.)   8    6
 ( 3., 3.)   8    8      A(I), IROW(I), ICOL(I), I=1,...,NNZ
 (  7., 11.)
 (  1., 24.)
 (-13.,-18.)
 (-10.,  3.)
 ( 23., 14.)
 ( 17., -7.)
 ( 15., -3.)
 ( -3., 20.)             B(I), I=1,...,N
 ( 0., 0.)
 ( 0., 0.)
 ( 0., 0.)
 ( 0., 0.)
 ( 0., 0.)
 ( 0., 0.)
 ( 0., 0.)
 ( 0., 0.)               X(I), I=1,...,N
```

9.3 Program Results

```
 F11DXF Example Program Results

 Monitoring at iteration no.   1
 residual norm:     1.5062E+02

 Monitoring at iteration no.   2
 residual norm:     1.5704E+02

 Monitoring at iteration no.   3
 residual norm:     1.4803E+02

 Monitoring at iteration no.   4
 residual norm:     8.5215E+01

 Monitoring at iteration no.   5
 residual norm:     4.2951E+01
```

```
 Monitoring at iteration no.   6
 residual norm:     2.5055E+01

 Monitoring at iteration no.   7
 residual norm:     1.9090E-01

 Final Results
 Number of iterations for convergence:       8
 Residual norm:                               3.3292E-08
 Right-hand side of termination criterion:    8.9100E-04
 1-norm of matrix A:                          2.7000E+01

    Solution vector
(       1.0000E-00,       1.0000E-00)
(       2.0000E+00,      -1.0000E-00)
(       3.0000E+00,       1.0000E-00)
(       4.0000E+00,      -1.0000E-00)
(       3.0000E+00,      -1.0000E-00)
(       2.0000E+00,       1.0000E+00)
(       1.0000E+00,      -1.0000E-00)
(      -9.5326E-10,       3.0000E+00)

    Residual vector
(      -1.2622E-09,       1.6809E-09)
(      -2.7422E-09,       3.6071E-09)
(       6.6013E-10,      -1.9902E-09)
(      -3.2883E-10,       3.7184E-09)
(       3.4339E-09,       2.1164E-09)
(       7.3777E-10,      -2.9778E-09)
(       3.4214E-10,       1.2555E-09)
(       3.6653E-09,       2.7728E-09)
```