NAG 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,       &
                   CHECK, B, X, DIAG, WORK, IFAIL)

INTEGER                 NITER, N, NNZ, IROW(NNZ), ICOL(NNZ), IFAIL
COMPLEX (KIND=nag_wp) 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) and Young (1971)):

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

where $k = 1, 2, \ldots,$NITER and $x_0 = 0$.

F11DXF can be used both for non-Hermitian and Hermitian systems of equations. For Hermitian matrices, either all nonzero elements of the matrix A can be supplied using coordinate storage (CS), or only the nonzero 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).

STORE = 'N'
The complete matrix A is stored in CS format.

STORE = 'S'
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 $A^{\mathrm{H}}x = b$ is required.

TRANS = 'N'
The approximate solution of $Ax = b$ is calculated.

TRANS = 'T'
The approximate solution of $A^{\mathrm{H}}x = b$ is calculated.

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

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

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. If DIAG already contains the diagonal of A, it must be set to 'N'.

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

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

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

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

4: NITER – INTEGER *Input*

On entry: the number of Jacobi iterations requested.

Constraint: NITER ≥ 1.

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 nonzero elements in the matrix A.

If STORE = 'S', the number of nonzero elements in the lower triangle of the matrix A.

Constraints:

if STORE = 'N', $1 \leq \mathrm{NNZ} \leq \mathrm{N}^2$;
if STORE = 'S', $1 \leq \mathrm{NNZ} \leq \mathrm{N} \times (\mathrm{N} + 1)/2$.

7: A(NNZ) – COMPLEX (KIND=nag_wp) array *Input*

On entry: if STORE = 'N', the nonzero elements in the matrix A (CS format).

If STORE = 'S', the nonzero 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 nonzero elements supplied in A.

If STORE = 'S', the row and column indices of the nonzero elements of the lower triangle of the matrix A supplied in A.

Constraints:

$1 \leq \text{IROW}(i) \leq \text{N}$, for $i = 1, 2, \ldots, \text{NNZ}$;
if STORE = 'N', $1 \leq \text{ICOL}(i) \leq \text{N}$, for $i = 1, 2, \ldots, \text{NNZ}$;
if STORE = 'S', $1 \leq \text{ICOL}(i) \leq \text{IROW}(i)$, for $i = 1, 2, \ldots, \text{NNZ}$;
either $\text{IROW}(i-1) < \text{IROW}(i)$ or both $\text{IROW}(i-1) = \text{IROW}(i)$ 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 or SCS representation of the matrix A should be checked.

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

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

See also Section 9.2.

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

11: B(N) – COMPLEX (KIND=nag_wp) array *Input*

On entry: the right-hand side vector b.

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

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

13: DIAG(N) – COMPLEX (KIND=nag_wp) 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 (KIND=nag_wp) array *Workspace*

15: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction 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 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.**

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

IFAIL $= 3$

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

$1 \leq \mathrm{IROW}(i) \leq \mathrm{N}$ and

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

if STORE = 'S' then $1 \leq \mathrm{ICOL}(i) \leq \mathrm{IROW}(i)$, 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 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.

IFAIL $= -99$

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

See Section 3.8 in the Essential Introduction for further information.

IFAIL $= -399$

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

See Section 3.7 in the Essential Introduction for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

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 Parallelism and Performance

F11DXF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F11DXF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

9.1 Timing

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

9.2 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 = 'C' for the first of such calls, and to set CHECK = 'N' for all subsequent calls.

10 Example

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

10.1 Program Text

```
    Program f11dxfe

!     F11DXF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: f11brf, f11bsf, f11btf, f11dxf, f11xnf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: anorm, sigmax, stplhs, stprhs, tol
      Integer                          :: i, ifail, ifail1, irevcm, iterm,     &
                                          itn, lwork, lwreq, m, maxitn, monit, &
                                          n, niter, nnz
      Character (1)                    :: init, norm, precon, weight
```

```
      Character (8)                    :: method
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:), b(:), diag(:), work(:), x(:)
      Real (Kind=nag_wp), Allocatable  :: wgt(:)
      Integer, Allocatable             :: icol(:), irow(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: log, nint
!     .. Executable Statements ..
      Write (nout,*) 'F11DXF Example Program Results'

!     Skip heading in data file

      Read (nin,*)
      Read (nin,*) n
      Read (nin,*) nnz
      lwork = 300
      Allocate (a(nnz),b(n),diag(n),work(lwork),x(n),wgt(n),icol(nnz), &
        irow(nnz))

!     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_nag_wp
      sigmax = 0.0E0_nag_wp

!     Read the parameters for the preconditioner

      Read (nin,*) niter

!     Read the non-zero elements of the matrix A

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

!     Read right-hand side vector b and initial approximate solution

      Read (nin,*) b(1:n)
      Read (nin,*) x(1:n)

!     Call F11BDF to initialize the solver

!     ifail: behaviour on error exit
!             =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      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

      irevcm = 0
      init = 'I'

      ifail = 0
loop: Do
        Call f11bsf(irevcm,x,b,wgt,work,lwreq,ifail)

        If (irevcm/=4) Then
          ifail1 = -1
          If (irevcm==-1) Then
            Call f11xnf('Transpose',n,nnz,a,irow,icol,'No checking',x,b, &
              ifail1)
          Else If (irevcm==1) Then
            Call f11xnf('No transpose',n,nnz,a,irow,icol,'No checking',x,b, &
```

```
              ifail1)
          Else If (irevcm==2) Then
            Call f11dxf('Non Hermitian','N',init,niter,n,nnz,a,irow,icol, &
              'Check',x,b,diag,work(lwreq+1),ifail1)
            init = 'N'
          Else If (irevcm==3) Then
            Call f11btf(itn,stplhs,stprhs,anorm,sigmax,work,lwreq,ifail1)
            If (ifail1==0) Then
              Write (nout,99999) itn
              Write (nout,99998) nint(log(stplhs)/log(10.0_nag_wp))
            End If
          End If
          If (ifail1/=0) irevcm = 6
        Else
          Exit loop
        End If
      End Do loop

!     Obtain information about the computation

      ifail1 = 0
      Call f11btf(itn,stplhs,stprhs,anorm,sigmax,work,lwreq,ifail1)

!     Print the output data

      Write (nout,99997)
      Write (nout,99996) 'Number of iterations for convergence:    ', itn
      Write (nout,99995) 'Residual norm:                           ', stplhs
      Write (nout,99995) 'Right-hand side of termination criterion:', stprhs
      Write (nout,99995) '1-norm of matrix A:                      ', anorm

!     Output x

      Write (nout,99994)
      Write (nout,99993) x(1:n)

99999 Format (/1X,'Monitoring at iteration number',I4)
99998 Format (1X,'  order of residual norm:',I4)
99997 Format (/1X,'Final Results')
99996 Format (1X,A,I5)
99995 Format (1X,A,1P,E11.1)
99994 Format (/2X,'  Solution vector')
99993 Format (1X,'(',F8.3,',',F8.3,')')
    End Program f11dxfe
```

10.2 Program Data

```
F11DXF Example Program Data
  8                        : n
 24                        : nnz
  'TFQMR'                  : method
  'P'     '1'       'N'  1 : precon, norm, weight, iterm
  2       1.0D-12   20     : m, tol, maxitn
  1                        : monit
  2                        : niter

 (  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) 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(1:n)

(  0.,  0.)
(  0.,  0.)
(  0.,  0.)
(  0.,  0.)
(  0.,  0.)
(  0.,  0.)
(  0.,  0.)
(  0.,  0.)               : (Initial guess) x(1:n)
```

10.3 Program Results

```
F11DXF Example Program Results

Monitoring at iteration number   1
  order of residual norm:   2

Monitoring at iteration number   2
  order of residual norm:   2

Monitoring at iteration number   3
  order of residual norm:   2

Monitoring at iteration number   4
  order of residual norm:   2

Monitoring at iteration number   5
  order of residual norm:   2

Monitoring at iteration number   6
  order of residual norm:   1

Monitoring at iteration number   7
  order of residual norm:  -1

Monitoring at iteration number   8
  order of residual norm:  -7

Monitoring at iteration number   9
  order of residual norm:  -9

Final Results
Number of iterations for convergence:       10
Residual norm:                               9.5E-12
Right-hand side of termination criterion:    8.9E-10
1-norm of matrix A:                          2.7E+01

   Solution vector
(   1.000,   1.000)
(   2.000,  -1.000)
(   3.000,   1.000)
```

```
 (    4.000,  -1.000)
 (    3.000,  -1.000)
 (    2.000,   1.000)
 (    1.000,  -1.000)
 (   -0.000,   3.000)
```