NAG Library Routine Document

F11JDF

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

F11JDF solves a system of linear equations involving the preconditioning matrix corresponding to SSOR applied to a real sparse symmetric matrix, represented in symmetric coordinate storage format.

2 Specification

```
SUBROUTINE F11JDF (N, NNZ, A, IROW, ICOL, RDIAG, OMEGA, CHECK, Y, X,       &
                   IWORK, IFAIL)

INTEGER             N, NNZ, IROW(NNZ), ICOL(NNZ), IWORK(N+1), IFAIL
REAL (KIND=nag_wp) A(NNZ), RDIAG(N), OMEGA, Y(N), X(N)
CHARACTER(1)        CHECK
```

3 Description

F11JDF solves a system of equations

$$Mx = y$$

involving the preconditioning matrix

$$M = \frac{1}{\omega(2-\omega)}(D+\omega L)D^{-1}(D+\omega L)^{\mathrm{T}}$$

corresponding to symmetric successive-over-relaxation (SSOR) (see Young (1971)) on a linear system $Ax = b$, where A is a sparse symmetric matrix stored in symmetric coordinate storage (SCS) format (see Section 2.1.2 in the F11 Chapter Introduction).

In the definition of M given above D is the diagonal part of A, L is the strictly lower triangular part of A, and ω is a user-defined relaxation parameter.

It is envisaged that a common use of F11JDF will be to carry out the preconditioning step required in the application of F11GEF to sparse linear systems. For an illustration of this use of F11JDF see the example program given in Section 10.1. F11JDF is also used for this purpose by the Black Box routine F11JEF.

4 References

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

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 lower triangular part of A.

Constraint: $1 \leq \mathrm{NNZ} \leq \mathrm{N} \times (\mathrm{N}+1)/2$.

3: A(NNZ) – REAL (KIND=nag_wp) array *Input*

On entry: the nonzero elements in the lower triangular part of 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 F11ZBF may be used to order the elements in this way.

4: IROW(NNZ) – INTEGER array *Input*

5: ICOL(NNZ) – INTEGER array *Input*

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

Constraints:

IROW and ICOL must satisfy these constraints (which may be imposed by a call to F11ZBF):

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

6: RDIAG(N) – REAL (KIND=nag_wp) array *Input*

On entry: the elements of the diagonal matrix D^{-1}, where D is the diagonal part of A.

7: OMEGA – REAL (KIND=nag_wp) *Input*

On entry: the relaxation parameter ω.

Constraint: $0.0 < \text{OMEGA} < 2.0$.

8: CHECK – CHARACTER(1) *Input*

On entry: specifies whether or not the input data should be checked.

CHECK = 'C'
Checks are carried out on the values of N, NNZ, IROW, ICOL and OMEGA.

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

See also Section 9.2.

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

9: Y(N) – REAL (KIND=nag_wp) array *Input*

On entry: the right-hand side vector y.

10: X(N) – REAL (KIND=nag_wp) array *Output*

On exit: the solution vector x.

11: IWORK(N + 1) – INTEGER array *Workspace*

12: 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, CHECK $\neq$ 'C' or 'N'.

IFAIL $= 2$

On entry, N < 1,
or NNZ < 1,
or NNZ $>$ N $\times$ (N $+ 1)/2$,
or OMEGA lies outside the interval $(0.0, 2.0)$,

IFAIL $= 3$

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

$1 \leq \mathrm{IROW}(i) \leq \mathrm{N}$ and $1 \leq \mathrm{ICOL}(i) \leq \mathrm{IROW}(i)$, 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 in the lower triangular part of A, is out of order, or has duplicate row and column indices. Call F11ZBF to reorder and sum or remove duplicates.

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

$$|\delta M| \leq c(n)\epsilon |D + \omega L| \left|D^{-1}\right| \left|(D + \omega L)^{\mathrm{T}}\right|,$$

$c(n)$ is a modest linear function of n, and ϵ is the ***machine precision***.

8 Parallelism and Performance

F11JDF is not threaded in any implementation.

9 Further Comments

9.1 Timing

The time taken for a call to F11JDF is proportional to NNZ.

9.2 Use of CHECK

It is expected that a common use of F11JDF will be to carry out the preconditioning step required in the application of F11GEF to sparse symmetric linear systems. In this situation F11JDF 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 solves a sparse symmetric linear system of equations

$$Ax = b,$$

using the conjugate-gradient (CG) method with SSOR preconditioning.

The CG algorithm itself is implemented by the reverse communication routine F11GEF, which returns repeatedly to the calling program with various values of the argument IREVCM. This argument indicates the action to be taken by the calling program.

If IREVCM = 1, a matrix-vector product $v = Au$ is required. This is implemented by a call to F11XEF.

If IREVCM = 2, a solution of the preconditioning equation $Mv = u$ is required. This is achieved by a call to F11JDF.

If IREVCM = 4, F11GEF has completed its tasks. Either the iteration has terminated, or an error condition has arisen.

For further details see the routine document for F11GEF.

10.1 Program Text

```
    Program f11jdfe

!     F11JDF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: f11gdf, f11gef, f11gff, f11jdf, f11xef, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: anorm, omega, sigerr, sigmax,        &
                                          sigtol, stplhs, stprhs, tol
      Integer                          :: i, ifail, ifail1, irevcm, iterm,     &
                                          itn, its, liwork, lwneed, lwork,     &
                                          maxitn, maxits, monit, n, nnz
      Character (1)                    :: ckjdf, ckxef, norm, precon, sigcmp,  &
                                          weight
      Character (6)                    :: method
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:), b(:), rdiag(:), wgt(:),        &
                                          work(:), x(:)
      Integer, Allocatable             :: icol(:), irow(:), iwork(:)
!     .. Executable Statements ..
      Write (nout,*) 'F11JDF Example Program Results'
!     Skip heading in data file
```

```
      Read (nin,*)

!     Read algorithmic parameters

      Read (nin,*) n
      Read (nin,*) nnz
      liwork = n + 1
      lwork = 6*n + 120

      Allocate (a(nnz),b(n),rdiag(n),wgt(n),work(lwork),x(n),icol(nnz),        &
        irow(nnz),iwork(liwork))
      Read (nin,*) method
      Read (nin,*) precon, sigcmp, norm, iterm
      Read (nin,*) tol, maxitn
      Read (nin,*) anorm, sigmax
      Read (nin,*) sigtol, maxits
      Read (nin,*) omega

!     Read 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 x

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

!     Call F11GDF to initialize solver

      weight = 'N'
      monit = 0

!     ifail: behaviour on error exit
!             =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
      Call f11gdf(method,precon,sigcmp,norm,weight,iterm,n,tol,maxitn,anorm,   &
        sigmax,sigtol,maxits,monit,lwneed,work,lwork,ifail)

!     Calculate reciprocal diagonal matrix elements.

      iwork(1:n) = 0

      Do i = 1, nnz
        If (irow(i)==icol(i)) Then
          iwork(irow(i)) = iwork(irow(i)) + 1
          If (a(i)/=0.0E0_nag_wp) Then
            rdiag(irow(i)) = 1.0E0_nag_wp/a(i)
          Else
            Write (nout,*) 'Matrix has a zero diagonal element'
            Go To 100
          End If
        End If
      End Do

      Do i = 1, n
        If (iwork(i)==0) Then
          Write (nout,*) 'Matrix has a missing diagonal element'
          Go To 100
        End If
        If (iwork(i)>=2) Then
          Write (nout,*) 'Matrix has a multiple diagonal element'
          Go To 100
        End If
      End Do

!     Call F11GEF to solve the linear system

      irevcm = 0
      ckxef = 'C'
```

```
      ckjdf = 'C'

      ifail = 1
loop: Do
        Call f11gef(irevcm,x,b,wgt,work,lwork,ifail)

        If (irevcm/=4) Then
          ifail1 = -1
          Select Case (irevcm)
          Case (1)
!           Compute matrix vector product

            Call f11xef(n,nnz,a,irow,icol,ckxef,x,b,ifail1)

            ckxef = 'N'
          Case (2)
!           SSOR preconditioning

            Call f11jdf(n,nnz,a,irow,icol,rdiag,omega,ckjdf,x,b,iwork,ifail1)

            ckjdf = 'N'
          End Select
          If (ifail1/=0) Then
            irevcm = 6
          End If
        Else If (ifail/=0) Then
          Write (nout,99996) ifail
          Go To 100
        Else
          Exit loop
        End If
      End Do loop

!     Termination

      Call f11gff(itn,stplhs,stprhs,anorm,sigmax,its,sigerr,work,lwork,ifail)

      Write (nout,99999) 'Converged in', itn, ' iterations'
      Write (nout,99998) 'Final residual norm =', stplhs

!     Output x

      Write (nout,99997) x(1:n)

100   Continue

99999 Format (1X,A,I10,A)
99998 Format (1X,A,1P,E16.3)
99997 Format (1X,1P,E16.4)
99996 Format (1X,/,1X,' ** F11GEF returned with IFAIL = ',I5)
    End Program f11jdfe
```

10.2 Program Data

```
F11JDF Example Program Data
  7                   N
 16                   NNZ
 'CG'                 METHOD
 'P' 'N' 'I' 1        PRECON, SIGCMP, NORM, ITERM
 1.0D-6 100           TOL, MAXITN
 0.0D0 0.0D0          ANORM, SIGMAX
 0.0D0 10             SIGTOL, MAXITS
 1.0D0                OMEGA
  4.   1    1
  1.   2    1
  5.   2    2
  2.   3    3
  2.   4    2
  3.   4    4
 -1.   5    1
```

```
 1.   5    4
 4.   5    5
 1.   6    2
-2.   6    5
 3.   6    6
 2.   7    1
-1.   7    2
-2.   7    3
 5.   7    7         A(I), IROW(I), ICOL(I), I=1,...,NNZ
15.  18.  -8.   21.
11.  10.   29.       B(I), I=1,...,N
 0.   0.   0.    0.
 0.   0.   0.        X(I), I=1,...,N
```

10.3 Program Results

```
F11JDF Example Program Results
Converged in         6 iterations
Final residual norm =       7.105E-15
      1.0000E+00
      2.0000E+00
      3.0000E+00
      4.0000E+00
      5.0000E+00
      6.0000E+00
      7.0000E+00
```