NAG Library Routine Document

F11JRF

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

F11JRF solves a system of linear equations involving the preconditioning matrix corresponding to SSOR applied to a complex sparse Hermitian matrix, represented in symmetric coordinate storage format.

2 Specification

```
SUBROUTINE F11JRF (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)       RDIAG(N), OMEGA
COMPLEX (KIND=nag_wp) A(NNZ), Y(N), X(N)
CHARACTER(1)             CHECK
```

3 Description

F11JRF solves a system of equations

$$Mx = y$$

involving the preconditioning matrix

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

corresponding to symmetric successive-over-relaxation (SSOR) (see Young (1971)) on a linear system $Ax = b$, where A is a sparse complex Hermitian 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. Note that since A is Hermitian the matrix D is necessarily real.

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 the matrix A.

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

3: A(NNZ) – COMPLEX (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 F11ZPF 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 the following constraints (which may be imposed by a call to F11ZPF):

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

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. Note that since A is Hermitian the elements of D^{-1} are necessarily real.

7: OMEGA – REAL (KIND=nag_wp) *Input*

On entry: the relaxation parameter ω.

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

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.

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

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

On entry: the right-hand side vector y.

10: X(N) – COMPLEX (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:

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

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 F11ZPF to reorder and sum or remove duplicates.

IFAIL $= 4$

On entry, a row of A has no diagonal entry.

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

F11JRF is not threaded in any implementation.

9 Further Comments

9.1 Timing

The time taken for a call to F11JRF is proportional to NNZ.

10 Example

This example program solves the preconditioning equation $Mx = y$ for a 9 by 9 sparse complex Hermitian matrix A, given in symmetric coordinate storage (SCS) format.

10.1 Program Text

```
    Program f11jrfe

!     F11JRF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: f11jrf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter              :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)              :: omega
      Integer                         :: i, ifail, n, nnz
      Character (1)                   :: check
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:), x(:), y(:)
      Real (Kind=nag_wp), Allocatable  :: rdiag(:)
      Integer, Allocatable            :: icol(:), irow(:), iwork(:)
!     .. Intrinsic Procedures ..
      Intrinsic                       :: real
!     .. Executable Statements ..
      Write (nout,*) 'F11JRF Example Program Results'
!     Skip heading in data file
      Read (nin,*)

!     Read algorithmic parameters

      Read (nin,*) n
      Read (nin,*) nnz

      Allocate (a(nnz),x(n),y(n),rdiag(n),icol(nnz),irow(nnz),iwork(n+1))
      Read (nin,*) check
      Read (nin,*) omega

!     Read the matrix A

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

!     Read rhs vector y

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

!     Fill in the diagonal part

      Do i = 1, nnz
        If (irow(i)==icol(i)) Then
```

```
            rdiag(irow(i)) = 1.E0_nag_wp/real(a(i))
          End If
        End Do

!       Solve Mx = b using F11JRF

!       ifail: behaviour on error exit
!               =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        ifail = 0
        Call f11jrf(n,nnz,a,irow,icol,rdiag,omega,check,y,x,iwork,ifail)

!       Output x

        Write (nout,99999) x(1:n)

99999 Format (1X,'(',E16.4,',',E16.4,')')
    End Program f11jrfe
```

10.2 Program Data

```
F11JRF Example Program Data
  9                    N
  23                   NNZ
 'C'                   CHECK
  1.1                  OMEGA
 ( 6., 0.)   1    1
 (-1., 1.)   2    1
 ( 6., 0.)   2    2
 ( 0., 1.)   3    2
 ( 5., 0.)   3    3
 ( 5., 0.)   4    4
 ( 2.,-2.)   5    1
 ( 4., 0.)   5    5
 ( 1., 1.)   6    3
 ( 2., 0.)   6    4
 ( 6., 0.)   6    6
 (-4., 3.)   7    2
 ( 0., 1.)   7    5
 (-1., 0.)   7    6
 ( 6., 0.)   7    7
 (-1.,-1.)   8    4
 ( 0.,-1.)   8    6
 ( 9., 0.)   8    8
 ( 1., 3.)   9    1
 ( 1., 2.)   9    5
 (-1., 0.)   9    6
 ( 1., 4.)   9    8
 ( 9., 0.)   9    9     A(I), IROW(I), ICOL(I), I=1,...,NNZ
 (  8.,  54.)
 (-10., -92.)
 ( 25.,  27.)
 ( 26., -28.)
 ( 54.,  12.)
 ( 26., -22.)
 ( 47.,  65.)
 ( 71., -57.)
 ( 60.,  70.)          Y(I), I=1,...,N
```

10.3 Program Results

```
 F11JRF Example Program Results
 (      0.1098E+01,      0.5914E+01)
 (      0.2230E+00,     -0.1408E+02)
 (      0.2232E+01,      0.7087E+01)
 (      0.4816E+01,     -0.6181E+01)
 (      0.6763E+01,      0.1569E+01)
 (      0.3353E+01,     -0.4785E+01)
 (      0.6699E+00,     -0.1465E+01)
 (      0.8832E+01,     -0.3633E+01)
 (      0.4768E+01,      0.1213E+00)
```