NAG Library Routine Document

F11DRF

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

F11DRF solves a system of linear equations involving the preconditioning matrix corresponding to SSOR applied to a complex sparse non-Hermitian matrix, represented in coordinate storage format.

2 Specification

```
SUBROUTINE F11DRF (TRANS, N, NNZ, A, IROW, ICOL, RDIAG, OMEGA, CHECK, Y,    &
                   X, IWORK, IFAIL)

INTEGER                 N, NNZ, IROW(NNZ), ICOL(NNZ), IWORK(2*N+1),         &
                        IFAIL
REAL (KIND=nag_wp)      OMEGA
COMPLEX (KIND=nag_wp)   A(NNZ), RDIAG(N), Y(N), X(N)
CHARACTER(1)            TRANS, CHECK
```

3 Description

F11DRF solves a system of linear equations

$$Mx = y, \quad \text{or} \quad M^{\mathrm{H}}x = y,$$

according to the value of the argument TRANS, where the matrix

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

corresponds to symmetric successive-over-relaxation (SSOR) Young (1971) applied to a linear system $Ax = b$, where A is a complex sparse non-Hermitian matrix stored in coordinate storage (CS) format (see Section 2.1.1 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, U is the strictly upper triangular part of A, and ω is a user-defined relaxation parameter.

It is envisaged that a common use of F11DRF will be to carry out the preconditioning step required in the application of F11BSF to sparse linear systems. For an illustration of this use of F11DRF see the example program given in Section 10. F11DRF is also used for this purpose by the Black Box routine F11DSF.

4 References

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

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'
$M^{\mathrm{H}}x = y$ is solved.

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

2: N – INTEGER *Input*

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

Constraint: N ≥ 1.

3: NNZ – INTEGER *Input*

On entry: the number of nonzero elements in the matrix A.

Constraint: $1 \leq \mathrm{NNZ} \leq \mathrm{N}^2$.

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

On entry: the nonzero elements in 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 F11ZNF may be used to order the elements in this way.

5: IROW(NNZ) – INTEGER array *Input*
6: ICOL(NNZ) – INTEGER array *Input*

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

Constraints:

IROW and ICOL must satisfy the following constraints (which may be imposed by a call to F11ZNF):

$1 \leq \mathrm{IROW}(i) \leq \mathrm{N}$ and $1 \leq \mathrm{ICOL}(i) \leq \mathrm{N}$, for $i = 1, 2, \ldots, \mathrm{NNZ}$;
either $\mathrm{IROW}(i-1) < \mathrm{IROW}(i)$ or both $\mathrm{IROW}(i-1) = \mathrm{IROW}(i)$ and $\mathrm{ICOL}(i-1) < \mathrm{ICOL}(i)$, for $i = 2, 3, \ldots, \mathrm{NNZ}$.

7: RDIAG(N) – COMPLEX (KIND=nag_wp) array *Input*

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

8: OMEGA – REAL (KIND=nag_wp) *Input*

On entry: the relaxation parameter ω.

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

9: 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, NNZ, IROW, ICOL and OMEGA.

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

See also Section 9.2.

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

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

On entry: the right-hand side vector y.

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

On exit: the solution vector x.

12: IWORK$(2 \times \mathrm{N} + 1)$ – INTEGER array *Workspace*

13: 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,
or NNZ < 1,
or NNZ $>$ N^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 $1 \leq \mathrm{ICOL}(i) \leq \mathrm{N}$, 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 matrix A, is out of order, or has duplicate row and column indices. Call F11ZNF to reorder and sum or remove duplicates.

IFAIL $= 4$

On entry, the matrix A has a zero diagonal element. The SSOR preconditioner is not appropriate for this problem.

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

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

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

8 Parallelism and Performance

F11DRF is not threaded in any implementation.

9 Further Comments

9.1 Timing

The time taken for a call to F11DRF is proportional to NNZ.

9.2 Use of CHECK

It is expected that a common use of F11DRF will be to carry out the preconditioning step required in the application of F11BSF to sparse linear systems. In this situation F11DRF 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 CHECK = 'N' for all subsequent calls.

10 Example

This example solves a complex sparse linear system of equations

$$Ax = b,$$

using RGMRES with SSOR preconditioning.

The RGMRES algorithm itself is implemented by the reverse communication routine F11BSF, 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 F11XNF.

If IREVCM $= -1$, a conjugate transposed matrix-vector product $v = A^{\mathrm{H}}u$ is required in the estimation of the norm of A. This is implemented by a call to F11XNF.

If IREVCM $= 2$, a solution of the preconditioning equation $Mv = u$ is required. This is achieved by a call to F11DRF.

If IREVCM $= 4$, F11BSF has completed its tasks. Either the iteration has terminated, or an error condition has arisen.

For further details see the routine document for F11BSF.

10.1 Program Text

```
    Program f11drfe

!     F11DRF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: f11brf, f11bsf, f11btf, f11drf, f11xnf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: anorm, omega, sigmax, stplhs,        &
                                          stprhs, tol
      Integer                          :: i, ifail, ifail1, irevcm, iterm,     &
                                          itn, liwork, lwneed, lwork, m,       &
                                          maxitn, monit, n, nnz
      Character (1)                    :: ckdrf, ckxnf, norm, precon, trans,   &
                                          weight
      Character (8)                    :: method
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:), b(:), rdiag(:), work(:),     &
                                          x(:)
      Real (Kind=nag_wp), Allocatable  :: wgt(:)
      Integer, Allocatable             :: icol(:), irow(:), iwork(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: max
!     .. Executable Statements ..
      Write (nout,*) 'F11DRF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)

!     Read algorithmic parameters

      Read (nin,*) n, m
      Read (nin,*) nnz
      lwork = max(121+n*(3+m)+m*(m+5),120+7*n,120+(2*n+m)*(m+2)+2*n,120+10*n)
      liwork = 2*n + 1
      Allocate (a(nnz),b(n),rdiag(n),work(lwork),x(n),wgt(n),icol(nnz),        &
        irow(nnz),iwork(liwork))
      Read (nin,*) method
      Read (nin,*) precon, norm, iterm
      Read (nin,*) tol, maxitn
      Read (nin,*) anorm, sigmax
      Read (nin,*) omega

!     Read the matrix A

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

!     Read rhs vector b and initial approximate solution x

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

!     Call F11BRF 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 f11brf(method,precon,norm,weight,iterm,n,m,tol,maxitn,anorm,sigmax, &
        monit,lwneed,work,lwork,ifail)
```

```
!     Calculate reciprocal diagonal matrix elements if necessary

      If (precon=='P' .Or. precon=='p') Then

        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,0.0E0_nag_wp)) Then
              rdiag(irow(i)) = (1.0E0_nag_wp,0.0E0_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

      End If

!     Call F11BSF to solve the linear system

      irevcm = 0
      ckxnf = 'C'
      ckdrf = 'C'

      ifail = 1
loop: Do
        Call f11bsf(irevcm,x,b,wgt,work,lwork,ifail)

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

            Call f11xnf(trans,n,nnz,a,irow,icol,ckxnf,x,b,ifail1)

            ckxnf = 'N'
          Case (-1)
!           Compute conjugate transposed matrix-vector product
            trans = 'T'

            Call f11xnf(trans,n,nnz,a,irow,icol,ckxnf,x,b,ifail1)

            ckxnf = 'N'
          Case (2)
!           SSOR preconditioning
            trans = 'N'

            Call f11drf(trans,n,nnz,a,irow,icol,rdiag,omega,ckdrf,x,b,iwork,   &
              ifail1)

            ckdrf = 'N'
          End Select
          If (ifail1/=0) Then
            irevcm = 6
          End If
```

```
        Else If (ifail==0) Then
!         Termination

          ifail = 0
          Call f11btf(itn,stplhs,stprhs,anorm,sigmax,work,lwork,ifail)

          Write (nout,99996) itn
          Write (nout,99997) 'Matrix norm =', anorm
          Write (nout,99997) 'Final residual norm =', stplhs
          Write (nout,*)

!         Output x
          Write (nout,*) '                   X'
          Write (nout,99998) x(1:n)

          Exit loop
        Else
          Write (nout,99999) ifail
          Exit loop
        End If
      End Do loop
100   Continue

99999 Format (1X,/,1X,' ** F11BSF returned with IFAIL = ',I5)
99998 Format (1X,'(',1P,E16.4,',',1P,E16.4,')')
99997 Format (1X,A,1P,E16.3)
99996 Format (1X,'Converged in',I10,' iterations')
    End Program f11drfe
```

10.2 Program Data

```
F11DRF Example Program Data
  5   2                 N, M
 16                     NNZ
 'CGS'                  METHOD
 'P' 'I' 1              PRECON, NORM, ITERM
 1.D-10 1000            TOL, MAXITN
 0.D0 0.D0              ANORM, SIGMAX
 1.4D0                  OMEGA
 ( 2., 3.)   1    1
 ( 1.,-1.)   1    2
 (-1., 0.)   1    4
 ( 0., 2.)   2    2
 (-2., 1.)   2    3
 ( 1., 0.)   2    5
 ( 0.,-1.)   3    1
 ( 5., 4.)   3    3
 ( 3.,-1.)   3    4
 ( 1., 0.)   3    5
 (-2., 2.)   4    1
 (-3., 1.)   4    4
 ( 0., 3.)   4    5
 ( 4.,-2.)   5    2
 (-2., 0.)   5    3
 (-6., 1.)   5    5      A(I), IROW(I), ICOL(I), I=1,...,NNZ
 ( -3., 3.)
 (-11., 5.)
 ( 23.,48.)
 (-41., 2.)
 (-28.,-31.)            B(I), I=1,...,N
 ( 0., 0.)
 ( 0., 0.)
 ( 0., 0.)
 ( 0., 0.)
 ( 0., 0.)              X(I), I=1,...,N
```

10.3 Program Results

```
 F11DRF Example Program Results

 Converged in         5 iterations
 Matrix norm =       1.500E+01
 Final residual norm =       1.776E-14

                     X
(      1.0000E+00,      2.0000E+00)
(      2.0000E+00,      3.0000E+00)
(      3.0000E+00,      4.0000E+00)
(      4.0000E+00,      5.0000E+00)
(      5.0000E+00,      6.0000E+00)
```