F11DDF – NAG Fortran Library Routine Document

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

F11DDF solves a system of linear equations involving the preconditioning matrix corresponding to SSOR applied to a real sparse nonsymmetric matrix, represented in coordinate storage format.

2 Specification

```
SUBROUTINE F11DDF(TRANS, N, NNZ, A, IROW, ICOL, RDIAG, OMEGA,
1                 CHECK, Y, X, IWORK, IFAIL)
INTEGER           N, NNZ, IROW(NNZ), ICOL(NNZ), IWORK(2*N+1), IFAIL
real              A(NNZ), RDIAG(N), OMEGA, Y(N), X(N)
CHARACTER*1       TRANS, CHECK
```

3 Description

This routine solves a system of linear equations

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

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) [1] applied to a linear system $Ax = b$, where A is a real sparse nonsymmetric matrix stored in coordinate storage (CS) format (see Section 2.1.1 of the 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 F11DDF will be to carry out the preconditioning step required in the application of F11BBF to sparse linear systems. For an illustration of this use of F11DDF see the example program given in Section 9. F11DDF is also used for this purpose by the black-box routine F11DEF.

4 References

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

5 Parameters

1: TRANS — CHARACTER*1 *Input*

On entry: specifies whether or not the matrix M is transposed:

if TRANS = 'N', then $Mx = y$ is solved;
if TRANS = 'T', then $M^T x = y$ 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 non-zero elements in the matrix A.

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

4: A(NNZ) — ***real*** array *Input*

On entry: the non-zero 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 F11ZAF 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 non-zero elements supplied in A.

Constraints: IROW and ICOL must satisfy the following constraints (which may be imposed by a call to F11ZAF):

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

7: RDIAG(N) — ***real*** array *Input*

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

8: OMEGA — ***real*** *Input*

On entry: the relaxation parameter ω.

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

9: CHECK — CHARACTER*1 *Input*

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

if CHECK = 'C', checks are carried on the values of N, NNZ, IROW, ICOL and OMEGA;
if CHECK = 'N', none of these checks are carried out.

See also Section 8.2.

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

10: Y(N) — ***real*** array *Input*

On entry: the right-hand side vector y.

11: X(N) — ***real*** array *Output*

On exit: the solution vector x.

12: IWORK(2∗N+1) — INTEGER array *Workspace*

13: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6 Errors 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 detected by the routine:

IFAIL = 1

On entry, TRANS ≠ 'N' or 'T',
or CHECK ≠ '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 $i = 1, 2, \ldots,$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 $i = 2, 3, \ldots,$NNZ.

Therefore a non-zero element has been supplied which does not lie in the matrix A, is out of order, or has duplicate row and column indices. Call F11ZAF 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.

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

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

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

8 Further Comments

8.1 Timing

The time taken for a call to F11DDF is proportional to NNZ.

8.2 Use of CHECK

It is expected that a common use of F11DDF will be to carry out the preconditioning step required in the application of F11BBF to sparse linear systems. In this situation F11DDF 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 to 'C' for the first of such calls, and to 'N' for all subsequent calls.

9 Example

This example program solves a sparse linear system of equations:

$$Ax = b,$$

using RGMRES with SSOR preconditioning.

The RGMRES algorithm itself is implemented by the reverse communication routine F11BBF, which returns repeatedly to the calling program with various values of the parameter IREVCM. This parameter 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 F11XAF.

If IREVCM = -1 a transposed matrix-vector product $v = A^T u$ is required in the estimation of the norm of A. This is implemented by a call to F11XAF.

If IREVCM = 2 a solution of the preconditioning equation $Mv = u$ is required. This is achieved by a call to F11DDF.

If IREVCM = 4 F11BBF has completed its tasks. Either the iteration has terminated, or an error condition has arisen.

For further details see the routine document for F11BBF.

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.

```
*     F11DDF Example Program Text
*     Mark 19 Revised. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LA, LIWORK, LWORK
      PARAMETER        (NMAX=1000,LA=10000,LIWORK=2*NMAX+1,LWORK=10000)
*     .. Local Scalars ..
      real             ANORM, OMEGA, SIGMAX, STPLHS, STPRHS, TOL
      INTEGER          I, IFAIL, IREVCM, ITERM, ITN, LWNEED, LWREQ, M,
     +                 MAXITN, MONIT, N, NNZ
      CHARACTER        CKDDF, CKXAF, NORM, PRECON, TRANS, WEIGHT
      CHARACTER*8      METHOD
*     .. Local Arrays ..
      real             A(LA), B(NMAX), RDIAG(NMAX), WGT(NMAX),
     +                 WORK(LWORK), X(NMAX)
      INTEGER          ICOL(LA), IROW(LA), IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         F11BDF, F11BEF, F11BFF, F11DDF, F11XAF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11DDF Example Program Results'
      WRITE (NOUT,*)
*     Skip heading in data file
      READ (NIN,*)
*
```

```
*     Read algorithmic parameters
*
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
         READ (NIN,*) NNZ
         READ (NIN,*) METHOD
         READ (NIN,*) PRECON, NORM, ITERM
         READ (NIN,*) M, TOL, MAXITN
         READ (NIN,*) ANORM, SIGMAX
         READ (NIN,*) OMEGA
*
*        Check size of workspace
*
         LWREQ = MAX(N*(M+3)+M*(M+5)+101,7*N+100,(2*N+M)*(M+2)+N+100,
     +           10*N+100)
         IF (LWORK.LT.LWREQ) THEN
            WRITE (NOUT,*) 'LWORK must be at least', LWREQ
            STOP
         END IF
*
*        Read the matrix A
*
         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 x
*
         READ (NIN,*) (B(I),I=1,N)
         READ (NIN,*) (X(I),I=1,N)
*
*        Call F11BDF to initialize solver
*
         WEIGHT = 'N'
         MONIT = 0
         IFAIL = 0
         CALL F11BDF(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.EQ.'P' .OR. PRECON.EQ.'p') THEN
*
            DO 40 I = 1, N
               IWORK(I) = 0
   40       CONTINUE
*
            DO 60 I = 1, NNZ
               IF (IROW(I).EQ.ICOL(I)) THEN
                  IWORK(IROW(I)) = IWORK(IROW(I)) + 1
                  IF (A(I).NE.0.e0) THEN
                     RDIAG(IROW(I)) = 1.e0/A(I)
                  ELSE
                     WRITE (NOUT,*) 'Matrix has a zero diagonal element'
                     GO TO 140
                  END IF
               END IF
   60       CONTINUE
```

```
*
            DO 80 I = 1, N
               IF (IWORK(I).EQ.0) THEN
                  WRITE (NOUT,*) 'Matrix has a missing diagonal element'
                  GO TO 140
               END IF
               IF (IWORK(I).GE.2) THEN
                  WRITE (NOUT,*)
     +              'Matrix has a multiple diagonal element'
                  GO TO 140
               END IF
   80       CONTINUE
*
         END IF
*
*        Call F11BEF to solve the linear system
*
         IREVCM = 0
         CKXAF = 'C'
         CKDDF = 'C'
*
  100    CONTINUE
*
         CALL F11BEF(IREVCM,X,B,WGT,WORK,LWORK,IFAIL)
*
         IF (IREVCM.EQ.1) THEN
*
*           Compute matrix-vector product
*
            TRANS = 'Not transposed'
            CALL F11XAF(TRANS,N,NNZ,A,IROW,ICOL,CKXAF,X,B,IFAIL)
            CKXAF = 'N'
            GO TO 100
*
         ELSE IF (IREVCM.EQ.-1) THEN
*
*           Compute transposed matrix-vector product
*
            TRANS = 'Transposed'
            CALL F11XAF(TRANS,N,NNZ,A,IROW,ICOL,CKXAF,X,B,IFAIL)
            CKXAF = 'N'
            GO TO 100
*
         ELSE IF (IREVCM.EQ.2) THEN
*
*           SSOR preconditioning
*
            TRANS = 'Not transposed'
            CALL F11DDF(TRANS,N,NNZ,A,IROW,ICOL,RDIAG,OMEGA,CKDDF,X,B,
     +                  IWORK,IFAIL)
            CKDDF = 'N'
            GO TO 100
*
         ELSE IF (IREVCM.EQ.4) THEN
*
```

```
*           Termination
*
            CALL F11BFF(ITN,STPLHS,STPRHS,ANORM,SIGMAX,WORK,LWORK,IFAIL)
*
            WRITE (NOUT,'(A,I10,A)') ' Converged in', ITN, ' iterations'
            WRITE (NOUT,'(A,1P,D16.3)') ' Matrix norm =', ANORM
            WRITE (NOUT,'(A,1P,D16.3)') ' Final residual norm =', STPLHS
            WRITE (NOUT,*)
*
*           Output x
*
            WRITE (NOUT,*) '           X'
            DO 120 I = 1, N
               WRITE (NOUT,'(1X,1P,D16.4)') X(I)
  120       CONTINUE
*
         END IF
*
  140    CONTINUE
*
      END IF
      STOP
      END
```

9.2 Program Data

```
F11DDF Example Program Data
  5                   N
 16                   NNZ
 'RGMRES'             METHOD
 'P' 'I' 1            PRECON, NORM, ITERM
 2 1.E-10 1000        M, TOL, MAXITN
 0.E0 0.E0            ANORM, SIGMAX
 1.1E0                OMEGA
  2.   1    1
  1.   1    2
 -1.   1    4
 -3.   2    2
 -2.   2    3
  1.   2    5
  1.   3    1
  5.   3    3
  3.   3    4
  1.   3    5
 -2.   4    1
 -3.   4    4
 -1.   4    5
  4.   5    2
 -2.   5    3
 -6.   5    5         A(I), IROW(I), ICOL(I), I=1,...,NNZ
  0.  -7.  33.
-19. -28.              B(I), I=1,...,N
  0.   0.   0.
  0.   0.              X(I), I=1,...,N
```

9.3 Program Results

```
F11DDF Example Program Results

Converged in        12 iterations
Matrix norm =       1.200E+01
Final residual norm =       3.841E-09

           X
      1.0000E+00
      2.0000E+00
      3.0000E+00
      4.0000E+00
      5.0000E+00
```