F01MAF – 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

F01MAF computes an incomplete Cholesky factorization of a real sparse symmetric positive-definite matrix A.

2 Specification

```
  SUBROUTINE F01MAF(N, NZ, A, LICN, IRN, LIRN, ICN, DROPTL, DENSW,
 1                  WKEEP, IKEEP, IWORK, ABORT, INFORM, IFAIL)
  INTEGER           N, NZ, LICN, IRN(LIRN), LIRN, ICN(LICN),
 1                  IKEEP(2*N), IWORK(6*N), INFORM(4), IFAIL
  real              A(LICN), DROPTL, DENSW, WKEEP(3*N)
  LOGICAL           ABORT(3)
```

3 Description

F01MAF computes an incomplete Cholesky factorization

$$C = PLDL^TP^T, \quad WAW = C + E$$

for the sparse symmetric positive-definite matrix A, where P is a permutation matrix, L is a unit lower triangular matrix, D is a diagonal matrix with positive diagonal elements, E is an error matrix representing elements dropped during the factorization and diagonal elements that have been modified to ensure that C is positive-definite, and W is a diagonal matrix, chosen to make the diagonal elements of WAW unity.

$W^{-1}CW^{-1}$ is a pre-conditioning matrix for A, and the factorization of C is intended to be used by F04MAF to solve systems of linear equations $Ax = b$.

The permutation matrix P is chosen to reduce the amount of fill-in that occurs in L and the user-supplied parameter DROPTL can also be used to control the amount of fill-in that occurs.

Full details on the factorization can be found in Munksgaard [1].

F01MAF is based on the Harwell Library routine MA31A.

4 References

[1] Munksgaard N (1980) Solving sparse symmetric sets of linear equations by pre-conditioned conjugate gradients *ACM Trans. Math. Software* **6** 206–219

5 Parameters

1: N — INTEGER *Input*

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

Constraint: N ≥ 1.

2: NZ — INTEGER *Input*

On entry: the number of non-zero elements in the upper triangular part of the matrix A, including the number of elements on the leading diagonal.

Constraint: NZ $\geq$ N.

3: A(LICN) — ***real*** array *Input/Output*

On entry: the first NZ elements of the array A must contain the non-zero elements of the upper triangular part of the sparse positive-definite symmetric matrix A, including the elements on the leading diagonal.

On exit: the first (NZ − N) elements of A contain the elements above the diagonal of the matrix WAW, where W is a diagonal matrix whose ith diagonal element is $w_i = a_{ii}^{-1/2}$. These elements are returned in order by rows and the value returned in ICN(k) gives the column index of the element returned in A(k). The value w_i is returned in the ith element of the array WKEEP. The next LROW elements of A, where LROW is the value returned in INFORM(1), return details of the factorization for use by F04MAF.

4: LICN — INTEGER *Input*

On entry: the dimension of the array A as declared in the (sub)program from which F01MAF is called. If fill-in is expected during the factorization, then a larger value of LICN will allow fewer elements to be dropped during the factorization, thus giving a more accurate factorization, which in turn will almost certainly mean that fewer iterations will be required by F04MAF.

Constraint: LICN $\geq 2 \times$ NZ.

5: IRN(LIRN) — INTEGER array *Input/Output*

On entry: IRN(k), for $k = 1, 2, \ldots,$ NZ must contain the row index of the non-zero element of the matrix A supplied in A(k).

On exit: the first LCOL elements of IRN, where LCOL is the value returned in INFORM(2), return details of the factorization for use by F04MAF.

6: LIRN — INTEGER *Input*

On entry: the dimension of the array IRN as declared in the (sub)program from which F01MAF is called. LIRN must be at least NZ, but, as with LICN, if fill-in is expected then a larger value of LIRN will allow a more accurate factorization. For this purpose LIRN should exceed NZ by the same amount that LICN exceeds $2 \times$ NZ.

Constraint: LIRN $\geq$ NZ.

7: ICN(LICN) — INTEGER array *Input/Output*

On entry: ICN(k), for $k = 1, 2, \ldots,$ NZ must contain the column index of the non-zero element of the matrix A supplied in A(k). Thus $a_{ij} =$ A(k), where $i =$ IRN(k) and $j =$ ICN(k).

On exit: the first (NZ − N) elements of ICN give the column indices of the first (NZ − N) elements returned in A. The next LROW elements of ICN return details of the factorization for use by F04MAF.

8: DROPTL — ***real*** *Input/Output*

On entry: a value in the range $[-1.0, 1.0]$ to be used as a tolerance in deciding whether or not to drop elements during the factorization. At the kth pivot step the element $a_{ij}^{(k+1)}$ is dropped if it would cause fill-in and if $|a_{ij}^{(k+1)}| <$ |DROPTL| $\times \sqrt{a_{ii}^{(k)} a_{jj}^{(k)}}$. If DROPTL is supplied as negative, then it is not altered during the factorization and so is unchanged on exit, but if DROPTL is supplied as positive then it may be altered by the routine with the aim of obtaining an accurate factorization in the space available. If DROPTL is supplied as -1.0, then no fill-in will occur during the factorization; and if DROPTL is supplied as 0.0 then a complete factorization is performed.

On exit: may be overwritten with the value used by the routine in order to obtain an accurate factorization in the space available, if DROPTL > 0.0 on entry.

9: DENSW — ***real*** *Input/Output*

On entry: a value in the range [0.0, 1.0] to be used in deciding whether or not to regard the active part of the matrix at the kth pivot step as being full. If the ratio of non-zero elements to the total number of elements is greater than or equal to DENSW, then the active part is regarded as full. If DENSW < 1.0, then the storage used is likely to increase compared to the case where DENSW $= 0$, but the execution time is likely to decrease.

Suggested value: DENSW $= 0.8$.

On exit: if on entry DENSW is not in the range [0.0, 1.0], then it is set to 0.8. Otherwise it is unchanged.

10: WKEEP(3∗N) — ***real*** array *Output*

On exit: information which must be passed unchanged to F04MAF. The first N elements contain the values w_i, for $i = 1, 2, \ldots, n$, and the next N elements contain the diagonal elements of D.

11: IKEEP(2∗N) — INTEGER array *Output*

On exit: information which must be passed unchanged to F04MAF.

12: IWORK(6∗N) — INTEGER array *Workspace*

13: ABORT(3) — LOGICAL array *Input*

On entry: if ABORT(1) = .TRUE., the routine will exit immediately on detecting duplicate elements and return IFAIL = 5. Otherwise when ABORT(1) = .FALSE., the calculations will continue using the sum of the duplicate entries. In either case details of the duplicate elements are output on the current advisory message unit (see X04ABF), unless suppressed by the value of IFAIL on entry.

If ABORT(2) = .TRUE., the routine will exit immediately on detecting a zero or negative pivot element and return IFAIL = 6. Otherwise when ABORT(2) = .FALSE., the zero or negative pivot element will be modified to ensure positive-definiteness and a message will be printed on the current advisory message unit, unless suppressed by the value of IFAIL on entry.

If ABORT(3) = .TRUE., the routine will exit immediately if the arrays A and ICN have been filled up and return IFAIL = 7. Otherwise when ABORT(3) = .FALSE., the data in the arrays is compressed to release more storage and a message will be printed on the current advisory message unit, unless suppressed by the value of IFAIL on entry. If DROPTL is positive on entry, it may be modified in order to allow a factorization to be completed in the available space.

Suggested values:

ABORT(1) = .TRUE.,
ABORT(2) = .TRUE.,
ABORT(3) = .TRUE..

14: INFORM(4) — INTEGER array *Output*

On exit: INFORM(1) returns the number of elements of A and ICN that have been used by the routine. Thus at least the first INFORM(1) elements of A and of ICN must be supplied to F04MAF.

Similarly, INFORM(2) returns the number of elements of IRN that have been used by the routine and so at least the first INFORM(2) elements must be supplied to F04MAF.

INFORM(3) returns the number of entries supplied in A that corresponded to diagonal and duplicate elements. If no duplicate entries were found, then INFORM(3) will return the value of N.

INFORM(4) returns the value k of the pivot step from which the active matrix was regarded as full.

INFORM must be passed unchanged to F04MAF.

15: IFAIL — INTEGER *Input/Output*

For this routine, the normal use of IFAIL is extended to control the printing of error and warning messages as well as specifying hard or soft failure (see Chapter P01).

Before entry, IFAIL must be set to a value with the decimal expansion cba, where each of the decimal digits c, b and a must have a value of 0 or 1.

$a = 0$ specifies hard failure, otherwise soft failure;
$b = 0$ suppresses error messages, otherwise error messages will be printed (see Section 6);
$c = 0$ suppresses warning messages, otherwise warning messages will be printed (see Section 6).

The recommended value for inexperienced users is 110 (i.e., hard failure with all messages printed).

Unless the routine detects an error (see Section 6), IFAIL contains 0 on exit.

6 Error Indicators and Warnings

For each error, an explanatory error message is output on the current error message unit (as defined by X04AAF), unless suppressed by the value of IFAIL on entry.

Errors detected by the routine:

IFAIL = 1

On entry, N < 1,
or NZ < N,
or LIRN < NZ,
or LICN < 2 × NZ.

IFAIL = 2

One of the conditions 0 < IRN(k) ≤ ICN(k) ≤ N is not satisfied so that A(k) is not in the upper triangle of the matrix. No further computation is attempted.

IFAIL = 3

One of the diagonal elements of the matrix A is zero or negative so that A is not positive-definite. No further computation is attempted.

IFAIL = 4

The available space has been used and no further compressions are possible. The user should either increase DROPTL, or allocate more space to A, IRN and ICN.

For all the remaining values of IFAIL the computations will continue in the case of soft failure, so that more than one advisory message may be printed.

IFAIL = 5

Duplicate elements have been detected and ABORT(1) = .TRUE..

IFAIL = 6

A zero or negative pivot element has been detected during the factorization and ABORT(2) = .TRUE..

This should not happen if A is an M-matrix (see Munksgaard [1]), but may occur for other types of positive-definite matrix.

IFAIL = 7

The available space has been used and ABORT(3) = .TRUE..

7 Accuracy

The accuracy of the factorization will be determined by the size of the elements that are dropped and the size of the modifications made to the diagonal elements. If these sizes are small then the computed factors will correspond to a matrix close to A and the number of iterations required by F04MAF will be small. The more incomplete the factorization, the higher the number of iterations required by F04MAF.

8 Further Comments

The time taken by the routine will depend upon the sparsity pattern of the matrix and the number of fill-ins that occur during the factorization. At the very least the time taken can be expected to be roughly proportional to $n\tau$, where τ is the number of non-zeros.

The routine is intended for use with positive-definite matrices, but the user is warned that it will not necessarily detect non-positive-definiteness. Indeed the routine may return a factorization that can satisfactorily be used by F04MAF even when A is not positive-definite, but this should not be relied upon as F04MAF may not converge.

9 Example

The example program illustrates the use of F01MAF in conjunction with F04MAF to solve the 16 linear equations $Ax = b$, where

$$b^T = \begin{pmatrix} \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \end{pmatrix}$$

The n by n matrix A arises in the solution of Laplace's equation in a unit square, using a 5-point formula with a 6 by 6 discretisation, with unity on the boundaries.

The drop tolerance, DROPTL, is taken as 0.1, and the density factor, DENSW, is taken as 0.8. The value IFAIL = 111 is used so that advisory and error messages will be printed, but soft failure would occur if IFAIL were returned as non-zero.

A relative accuracy of about 0.0001 is requested in the solution from F04MAF, with a maximum of 50 iterations.

The example program for F02FJF illustrates the use of F01MAF and F04MAF in solving an eigenvalue problem.

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.

```
*     F01MAF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          N, M, NN, LICN, LIRN
      PARAMETER        (N=4,M=4,NN=N*M,LICN=90,LIRN=50)
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      real             DENSW, DROPTL
      INTEGER          I, IFAIL, J, JJ, K, MM1, NZ
*     .. Local Arrays ..
      real             A(LICN), ACC(2), B(NN), WKEEP(NN,3), WORK(NN,3)
      INTEGER          ICN(LICN), IKEEP(NN,2), INFORM(4), IRN(LIRN),
     +                 IWORK(NN,6), NOITS(2)
      LOGICAL          ABORT(3)
*     .. External Subroutines ..
```

```
      EXTERNAL         F01MAF, F04MAF, X04ABF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F01MAF Example Program Results'
      CALL X04ABF(1,NOUT)
*
*     Initialise the right hand side vector B.
*
      MM1 = M - 1
      DO 40 I = 1, N
         K = (I-1)*M
         B(K+1) = 0.25e0
         DO 20 J = 2, MM1
            B(K+J) = 0.0e0
   20    CONTINUE
         B(K+M) = 0.25e0
   40 CONTINUE
      K = N*MM1
      DO 60 I = 1, M
         B(I) = B(I) + 0.25e0
         B(K+I) = B(K+I) + 0.25e0
   60 CONTINUE
*
*     Set up the coefficient matrix A.
*
      K = 0
      DO 80 I = 1, NN
         K = K + 1
         A(K) = 1.0e0
         IRN(K) = I
         ICN(K) = I
   80 CONTINUE
      DO 120 I = 1, N
         DO 100 J = 1, MM1
            K = K + 1
            JJ = (I-1)*N + J
            A(K) = -0.25e0
            IRN(K) = JJ
            ICN(K) = JJ + 1
  100    CONTINUE
  120 CONTINUE
      DO 140 I = N + 1, NN
         K = K + 1
         A(K) = -0.25e0
         IRN(K) = I - N
         ICN(K) = I
  140 CONTINUE
      NZ = K
*
*     Call F01MAF to perform the incomplete Cholesky factorization of
*     the sparse, symmetric, positive definite matrix A.
*
      DO 160 I = 1, 3
         ABORT(I) = .TRUE.
  160 CONTINUE
      DROPTL = 0.1e0
      DENSW = 0.8e0
      IFAIL = 111
*
```

```
      CALL F01MAF(NN,NZ,A,LICN,IRN,LIRN,ICN,DROPTL,DENSW,WKEEP,IKEEP,
     +            IWORK,ABORT,INFORM,IFAIL)
*
      WRITE (NOUT,*)
      IF (IFAIL.NE.0) THEN
         WRITE (NOUT,99999)
     +     ' Error IFAIL returned from F01MAF with value', IFAIL
      ELSE
         WRITE (NOUT,99999) ' No. of elements of A ( and ICN ) used is '
     +     , INFORM(1)
         WRITE (NOUT,99999) ' No. of elements of IRN used is           '
     +     , INFORM(2)
         WRITE (NOUT,*)
*
*        Call F04MAF to solve the sparse, symmetric, positive
*        definite system of linear equations  A*X = B.
*
         NOITS(1) = 50
         ACC(1) = 0.0001e0
         IFAIL = 111
*
         CALL F04MAF(NN,NZ,A,LICN,IRN,LIRN,ICN,B,ACC,NOITS,WKEEP,WORK,
     +               IKEEP,INFORM,IFAIL)
*
         IF (IFAIL.NE.0) THEN
            WRITE (NOUT,99999)
     +        ' Error IFAIL returned from F04MAF with value', IFAIL
         ELSE
            WRITE (NOUT,99999) ' Number of iterations = ', NOITS(2)
            WRITE (NOUT,99998) ' Norm of residual     = ', ACC(2)
            WRITE (NOUT,*)
            WRITE (NOUT,*) ' Solution vector'
            WRITE (NOUT,99997) (B(I),I=1,NN)
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,A,I4)
99998 FORMAT (1X,A,e12.3)
99997 FORMAT (1X,4e15.5)
      END
```

9.2 Program Data

None.

9.3 Program Results

```
 F01MAF Example Program Results

  No. of elements of A ( and ICN ) used is   28
  No. of elements of IRN used is             24

  Number of iterations =    3
  Norm of residual     =    0.836E-05

  Solution vector
     0.10000E+01    0.10000E+01    0.10000E+01    0.10000E+01
```

```
       0.10000E+01    0.10000E+01    0.10000E+01    0.10000E+01
       0.10000E+01    0.10000E+01    0.10000E+01    0.10000E+01
       0.10000E+01    0.10000E+01    0.10000E+01    0.10000E+01
```