F04MBF – 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

F04MBF solves a system of real sparse symmetric linear equations using a Lanczos algorithm.

2 Specification

```
      SUBROUTINE F04MBF(N, B, X, APROD, MSOLVE, PRECON, SHIFT, RTOL,
     1                  ITNLIM, MSGLVL, ITN, ANORM, ACOND, RNORM, XNORM,
     2                  WORK, RWORK, LRWORK, IWORK, LIWORK, INFORM, IFAIL)
      INTEGER           N, ITNLIM, MSGLVL, ITN, LRWORK, IWORK(LIWORK),
     1                  LIWORK, INFORM, IFAIL
      real              B(N), X(N), SHIFT, RTOL, ANORM, ACOND, RNORM,
     1                  XNORM, WORK(N,5), RWORK(LRWORK)
      LOGICAL           PRECON
      EXTERNAL          APROD, MSOLVE
```

3 Description

F04MBF solves the system of linear equations

$$(A - \lambda I)x = b \qquad (1)$$

where A is an n by n sparse symmetric matrix and λ is a scalar, which is of course zero if the solution of the equations

$$Ax = b$$

is required. It should be noted that neither A nor $(A - \lambda I)$ need be positive-definite.

λ is supplied as the parameter SHIFT, and allows F04MBF to be used for finding eigenvectors of A in methods such as Rayleigh quotient iteration (see for example Lewis [1]), in which case λ will be an approximation to an eigenvalue of A and b an approximation to an eigenvector of A.

The routine also provides an option to allow pre-conditioning and this will often reduce the number of iterations required by F04MBF.

F04MBF is based upon algorithm SYMMLQ (see Paige and Saunders [2]) and solves the equations by an algorithm based upon the Lanczos process. Details of the method are given in Paige and Saunders [2]. The routine does not require A explicitly, but A is specified via a user-supplied routine APROD which, given an n element vector c, must return the vector z given by

$$z = Ac.$$

The pre-conditioning option is based on the following reasoning. If A can be expressed in the form

$$A = I + B$$

where B is of rank ρ, then the Lanczos process converges (in exact arithmetic) in at most ρ iterations. If more generally A can be expressed in the form

$$A = M + C$$

where M is symmetric positive-definite and C has rank ρ, then

$$M^{-(1/2)}AM^{-(1/2)} = I + M^{-(1/2)}CM^{-(1/2)}$$

and $M^{-(1/2)}AM^{-(1/2)}$ also has rank ρ, and the Lanczos process applied to $M^{-(1/2)}AM^{-(1/2)}$ would again converge in at most ρ iterations. On a computer, the number of iterations may be greater than ρ, but the

Lanczos process may still be expected to converge rapidly. F04MBF does not require $M^{-(1/2)}AM^{-(1/2)}$ to be formed explicitly, but implicitly solves the equations

$$M^{-(1/2)}(A-\lambda I)M^{-(1/2)}y = M^{-(1/2)}b\ ,\ y = M^{1/2}x \tag{2}$$

with the user being required to supply a routine MSOLVE to solve the equations

$$Mz = c. \tag{3}$$

For the pre-conditioning option to be effective, it is desirable that equations (3) can be solved efficiently. The example program in Section 9 illustrates the use of this option.

If we let r denote the residual vector

$$r = b - (A-\lambda I)x$$

corresponding to an iterate x, then, when pre-conditioning has not been requested, the iterative procedure is terminated if it is estimated that

$$\|r\| \le tol.\|A-\lambda I\|.\|x\|, \tag{4}$$

where *tol* is a user-supplied tolerance, $\|r\|$ denotes the Euclidean length of the vector r and $\|A\|$ denotes the Frobenius (Euclidean) norm of the matrix A. When pre-conditioning has been requested, the iterative procedure is terminated if it is estimated that

$$\|M^{-(1/2)}r\| \le tol.\|M^{-(1/2)}(A-\lambda I)M^{-(1/2)}\|.\|M^{1/2}x\|. \tag{5}$$

Note that

$$M^{-(1/2)}r = (M^{-(1/2)}b) - M^{-(1/2)}(A-\lambda I)M^{-(1/2)}(M^{1/2}x)$$

so that $M^{-(1/2)}r$ is the residual vector corresponding to equation (2). The routine will also terminate if it is estimated that

$$\|A-\lambda I\|.\|x\| \ge \|b\|/\epsilon, \tag{6}$$

where ϵ is the ***machine precision***, when pre-conditioning has not been requested; or if it is estimated that

$$\|M^{-(1/2)}(A-\lambda I)M^{-(1/2)}\|.\|M^{1/2}x\| \ge \|M^{-(1/2)}b\|/\epsilon \tag{7}$$

when pre-conditioning has been requested. If (6) is satisfied then x is almost certainly an eigenvector of A corresponding to the eigenvalue λ. If λ was set to 0 (for the solution of $Ax = b$), then this condition simply means that A is effectively singular.

4 References

[1] Lewis J G (1977) Algorithms for sparse matrix eigenvalue problems *Technical Report STAN–CS–77–595* Computer Science Department, Stanford University

[2] Paige C C and Saunders M A (1975) Solution of sparse indefinite systems of linear equations *SIAM J. Numer. Anal.* **12** 617–629

5 Parameters

1: N — INTEGER *Input*

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

Constraint: N ≥ 1.

2: B(N) — ***real*** array *Input*

On entry: the right-hand side vector b.

3: X(N) — ***real*** array *Output*

On exit: the solution vector x.

4: APROD — SUBROUTINE, supplied by the user. *External Procedure*

APROD must return the vector $y = Ax$ for a given vector x.

Its specification is:

```
SUBROUTINE APROD(IFLAG, N, X, Y, RWORK, LRWORK, IWORK, LIWORK)
INTEGER          IFLAG, N, LRWORK, IWORK(LIWORK), LIWORK
real             X(N), Y(N), RWORK(LRWORK)
```

1: IFLAG — INTEGER *Input/Output*

On entry: IFLAG is always non-negative.

On exit: IFLAG may be used as a flag to indicate a failure in the computation of Ax.

If IFLAG is negative on exit from APROD, F04MBF will exit immediately with IFAIL set to IFLAG.

2: N — INTEGER *Input*

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

3: X(N) — ***real*** array *Input*

On entry: the vector x for which Ax is required.

4: Y(N) — ***real*** array *Output*

On exit: the vector $y = Ax$.

5: RWORK(LRWORK) — ***real*** array *User Workspace*
6: LRWORK — INTEGER *Input*
7: IWORK(LIWORK) — INTEGER array *User Workspace*
8: LIWORK — INTEGER *Input*

APROD is called from F04MBF with the parameters RWORK, LRWORK, IWORK and LIWORK as supplied to F04MBF. The user is free to use the arrays RWORK and IWORK to supply information to APROD and MSOLVE as an alternative to using COMMON.

APROD must be declared as EXTERNAL in the (sub)program from which F04MBF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

5: MSOLVE — SUBROUTINE, supplied by the user. *External Procedure*

MSOLVE is only referenced when PRECON is supplied as .TRUE.. When PRECON is supplied as .FALSE., then F04MBF may be called with APROD as the actual argument for MSOLVE. When PRECON is supplied as .TRUE., then MSOLVE must return the solution y of the equations M$y = x$ for a given vector x, where M must be symmetric positive-definite.

Its specification is:

```
SUBROUTINE MSOLVE(IFLAG, N, X, Y, RWORK, LRWORK, IWORK, LIWORK)
INTEGER          IFLAG, N, LRWORK, IWORK(LIWORK), LIWORK
real             X(N), Y(N), RWORK(LRWORK)
```

1: IFLAG — INTEGER *Input/Output*

On entry: IFLAG is always non-negative.

On exit: IFLAG may be used as a flag to indicate a failure in the solution of $My = x$.

If IFLAG is negative on exit from MSOLVE, F04MBF will exit immediately with IFAIL set to IFLAG.

2: N — INTEGER *Input*

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

3: X(N) — ***real*** array *Input*

On entry: the vector x for which the equations $My = x$ are to be solved.

4: Y(N) — ***real*** array *Output*

On exit: the solution to the equations $My = x$.

5: RWORK(LRWORK) — ***real*** array *User Workspace*
6: LRWORK — INTEGER *Input*
7: IWORK(LIWORK) — INTEGER array *User Workspace*
8: LIWORK — INTEGER *Input*

MSOLVE is called from F04MBF with the parameters RWORK, LRWORK, IWORK and LIWORK as supplied to F04MBF. The user is free to use the arrays RWORK and IWORK to supply information to APROD and MSOLVE as an alternative to using COMMON.

MSOLVE must be declared as EXTERNAL in the (sub)program from which F04MBF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

6: PRECON — LOGICAL *Input*

On entry: PRECON specifies whether or not pre-conditioning is required. If PRECON = .TRUE., then pre-conditioning will be invoked and MSOLVE will be referenced by F04MBF; if PRECON = .FALSE., then MSOLVE is not referenced.

7: SHIFT — ***real*** *Input*

On entry: the value of λ. If the equations $Ax = b$ are to be solved, then SHIFT must be supplied as zero.

8: RTOL — ***real*** *Input*

On entry: the tolerance for convergence, *tol*, of equation (4). RTOL should not normally be less than ϵ, where ϵ is the ***machine precision***.

9: ITNLIM — INTEGER *Input*

On entry: an upper limit on the number of iterations. If ITNLIM ≤ 0, then the value N is used in place of ITNLIM.

10: MSGLVL — INTEGER *Input*

On entry: the level of printing from F04MBF. If MSGLVL ≤ 0, then no printing occurs, but otherwise messages will be output on the advisory message channel (see X04ABF). A description of the printed output is given in Section 5.1 below. The level of printing is determined as follows:

MSGLVL ≤ 0

No printing.

MSGLVL = 1

A brief summary is printed just prior to return from F04MBF.

MSGLVL ≥ 2

A summary line is printed periodically to monitor the progress of F04MBF, together with a brief summary just prior to return from F04MBF.

11: ITN — INTEGER *Output*

On exit: the number of iterations performed.

12: ANORM — ***real*** *Output*

On exit: an estimate of $\|A-\lambda I\|$ when PRECON = .FALSE., and $\|M^{-(1/2)}(A-\lambda I)M^{-(1/2)}\|$ when PRECON = .TRUE..

13: ACOND — ***real*** *Output*

On exit: an estimate of the condition number of $(A - \lambda I)$ when PRECON = .FALSE., and of $M^{-(1/2)}(A - \lambda I)M^{-(1/2)}$ when PRECON = .TRUE.. This will usually be a substantial underestimate.

14: RNORM — ***real*** *Output*

On exit: $\|r\|$, where $r = b - (A - \lambda I)x$ and x is the solution returned in X.

15: XNORM — ***real*** *Output*

On exit: $\|x\|$, where x is the solution returned in X.

16: WORK(N,5) — ***real*** array *Workspace*

17: RWORK(LRWORK) — ***real*** array *User Workspace*

RWORK is not used by F04MBF, but is passed directly to routines APROD and MSOLVE and may be used to pass information to these routines.

18: LRWORK — INTEGER *Input*

On entry: the length of the array RWORK as declared in the (sub)program from which F04MBF is called.

Constraint: LRWORK ≥ 1.

19: IWORK(LIWORK) — INTEGER array *User Workspace*

IWORK is not used by F04MBF, but is passed directly to routines APROD and MSOLVE and may be used to pass information to these routines.

20: LIWORK — INTEGER *Input*

On entry: the length of the array IWORK as declared in the (sub)program from which F04MBF is called.

Constraint: LIWORK ≥ 1.

21: INFORM — INTEGER *Output*

On exit: the reason for termination of F04MBF as follows:

INFORM = 0

The right-hand side vector $b = 0$ so that the exact solution is $x = 0$. No iterations are performed in this case.

INFORM = 1

The termination criterion of equation (4) has been satisfied with *tol* as the value supplied in RTOL.

INFORM = 2

The termination criterion of equation (4) has been satisfied with *tol* equal to ϵ, where ϵ is the ***machine precision***. The value supplied in RTOL must have been less than ϵ and was too small for the machine.

INFORM = 3

The termination criterion of equation (5) has been satisfied so that X is almost certainly an eigenvector of A corresponding to the eigenvalue SHIFT.

The values INFORM = 4 and INFORM = 5 correspond to failure with IFAIL = 3 or IFAIL = 2 respectively (see Section 6) and when IFAIL is negative, INFORM will be set to the same negative value.

22: 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).

5.1 Description of the Printed Output

When MSGLVL > 0, then F04MBF will produce output (except in the case where the routine fails with IFAIL = 1) on the advisory message channel (see X04ABF).

The following notation is used in the output.

Output	**Meaning**
`RBAR`	$M^{-(1/2)}(b-(A-\lambda I)x)=\bar{r}$
`ABAR`	$M^{-(1/2)}(A-\lambda I)M^{-(1/2)}=\bar{A}$
`Y`	$M^{1/2}x$
`R`	$b-(A-\lambda I)x$
`NORM(A)`	$\|A\|$

Of course, when pre-conditioning has not been requested then the first three reduce to $(b-(A-\lambda I)x)$, $(A-\lambda I)$ and x respectively. When MSGLVL $\geq$ 2 then some initial information is printed and the following notation is used.

Output	**Meaning**
`BETA1`	$(b^TM^{-1}b)^{1/2}\equiv\beta_1$
`ALFA1`	$(1/\beta_1)^2(M^{-(1/2)}b)^T(M^{-(1/2)}AM^{-(1/2)})(M^{-(1/2)}b)\equiv\alpha_1$

and a summary line is printed periodically giving the following information:

Output	**Meaning**
`ITN`	Iteration number, k.
`X1(LQ)`	The first element of the vector x_k^L, where x_k^L is the current iterate. See Paige and Saunders [2] for details.
`X1(CG)`	The first element of the vector x_k^C, where x_k^C is the vector that would be obtained by conjugate gradients. See Paige and Saunders [2] for details.
`NORM(RBAR)`	$\|\bar{r}\|$, where $\bar{r}$ is as defined above and x is either x_k^L or x_k^C depending upon which is the best current approximation to the solution. (See LQ/CG below).
`NORM(T)`	The value $\|T_k\|$, where T_k is the tridiagonal matrix of the Lanczos process. This increases monotonically and is a lower bound on $\|\bar{A}\|$.
`COND(L)`	A monotonically increasing lower bound on the condition number of $\bar{A}$, $\|\bar{A}\|\|(\bar{A})^{-1}\|$.
`LQ/CG`	L is printed if x_k^L is the best current approximation to the solution and C is printed otherwise.

6 Error Indicators and Warnings

Errors detected by the routine:

IFAIL < 0

A negative value of IFAIL indicates an exit from F04MBF because the user has set IFLAG negative in APROD or MSOLVE. The value of IFAIL will be the same as the user's setting of IFLAG.

IFAIL = 1

On entry, N < 1,
or LRWORK < 1,
or LIWORK < 1.

IFAIL = 2

The pre-conditioning matrix M does not appear to be positive-definite. The user should check that MSOLVE is working correctly.

IFAIL = 3

The limit on the number of iterations has been reached. If IFAIL = 1 on entry then the latest approximation to the solution is returned in X and the values ANORM, ACOND, RNORM and XNORM are also returned.

The value of INFORM contains additional information about the termination of the routine and users must examine INFORM to judge whether the routine has performed successfully for the problem in hand. In particular INFORM = 3 denotes that the matrix $A-\lambda I$ is effectively singular: if the purpose of calling F04MBF is to solve a system of equations $Ax = b$, then this condition must be regarded as a failure, but if the purpose is to compute an eigenvector, this result would be very satisfactory.

7 Accuracy

The computed solution x will satisfy the equation

$$r = b - (A - \lambda I)x$$

where the value $\|r\|$ is as returned in the parameter RNORM.

8 Further Comments

The time taken by the routine is likely to be principally determined by the time taken in APROD and, when pre-conditioning has been requested, in MSOLVE. Each of these routines is called once every iteration.

The time taken by the remaining operations in F04MBF is approximately proportional to n.

9 Example

To solve the 10 equations $Ax = b$ given by

$$A = \begin{pmatrix} 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \\ 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 1 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 \end{pmatrix}, \quad b = \begin{pmatrix} 6 \\ 4 \\ 4 \\ 4 \\ 4 \\ 4 \\ 4 \\ 4 \\ 4 \\ 6 \end{pmatrix}.$$

The tridiagonal part of A is positive-definite and such tridiagonal equations can be solved efficiently by F04FAF. The form of A suggests that this tridiagonal part is a good candidate for the pre-conditioning matrix M and so we illustrate the use of F04MBF by pre-conditioning with the 10 by 10 matrix

$$M = \begin{pmatrix} 2 & 1 & 0 & \ldots & 0 \\ 1 & 2 & 1 & \ldots & 0 \\ 0 & 1 & 2 & \ldots & 0 \\ . & . & . & & . \\ . & . & . & & . \\ . & . & . & & . \\ 0 & 0 & 0 & \ldots & 2 \end{pmatrix}.$$

Since $A-M$ has only 2 non-zero elements and is obviously of rank 2, we can expect F04MBF to converge very quickly in this example. Of course, in practical problems we shall not usually be able to make such a good choice of M.

The example sets the tolerance RTOL $= 10^{-5}$.

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.

```
*     F04MBF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          N, LRWORK, LIWORK
      PARAMETER        (N=10,LRWORK=1,LIWORK=1)
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Scalars in Common ..
      INTEGER          JOB
*     .. Arrays in Common ..
      real             D(N), E(N)
*     .. Local Scalars ..
      real             ACOND, ANORM, RNORM, RTOL, SHFT, XNORM
      INTEGER          I, IFAIL, INFORM, ITN, ITNLIM, MSGLVL
      LOGICAL          PRECON
*     .. Local Arrays ..
      real             B(N), RWORK(LRWORK), WORK(N,5), X(N)
      INTEGER          IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         APROD, F04MBF, MSOLVE, X04ABF
*     .. Common blocks ..
      COMMON           /USER/D, E, JOB
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F04MBF Example Program Results'
      WRITE (NOUT,*)
      CALL X04ABF(1,NOUT)
*
*     Set up the matrix M with the diagonal elements in D and
*     the off-diagonal elements in E.
*
      D(1) = 2.0e0
      DO 20 I = 2, N
         D(I) = 2.0e0
         E(I) = 1.0e0
   20 CONTINUE
*
*     Set JOB to zero so that F04FAF factorizes M on the first call
*     inside MSOLVE.
*
      JOB = 0
*
*     Initialize RHS and other quantities required by F04MBF.
*
      B(1) = 6.0e0
      DO 40 I = 2, N - 1
         B(I) = 4.0e0
   40 CONTINUE
      B(N) = 6.0e0
      PRECON = .TRUE.
      SHFT = 0.0e0
      RTOL = 0.00001e0
      ITNLIM = 100
*     * Set MSGLVL to 2 to get output at each iteration *
      MSGLVL = 1
```

```
      IFAIL = 1
*
      CALL F04MBF(N,B,X,APROD,MSOLVE,PRECON,SHFT,RTOL,ITNLIM,MSGLVL,ITN,
     +            ANORM,ACOND,RNORM,XNORM,WORK,RWORK,LRWORK,IWORK,
     +            LIWORK,INFORM,IFAIL)
*
      WRITE (NOUT,*)
      IF (IFAIL.NE.0) THEN
         WRITE (NOUT,99999) 'F04MBF fails. IFAIL =', IFAIL
      ELSE
         WRITE (NOUT,*) 'Solution returned by F04MBF'
         WRITE (NOUT,99998) (X(I),I=1,N)
      END IF
      STOP
*
99999 FORMAT (1X,A,I3)
99998 FORMAT (1X,5F9.3)
      END
*
      SUBROUTINE APROD(IFLAG,N,X,Y,RWORK,LRWORK,IWORK,LIWORK)
*     APROD  returns Y = A*X for a given X.
*     .. Scalar Arguments ..
      INTEGER          IFLAG, LIWORK, LRWORK, N
*     .. Array Arguments ..
      real             RWORK(LRWORK), X(N), Y(N)
      INTEGER          IWORK(LIWORK)
*     .. Local Scalars ..
      INTEGER          I
*     .. Executable Statements ..
      Y(1) = 2.0e0*X(1) + X(2) + 3.0e0*X(N)
      DO 20 I = 2, N - 1
         Y(I) = X(I-1) + 2.0e0*X(I) + X(I+1)
   20 CONTINUE
      Y(N) = 3.0e0*X(1) + X(N-1) + 2.0e0*X(N)
      RETURN
      END
*
      SUBROUTINE MSOLVE(IFLAG,N,X,Y,RWORK,LRWORK,IWORK,LIWORK)
*     Given X, MSOLVE solves the equations M*Y = X for Y, without
*     altering X, by calling the NAG Library routine F04FAF.
*     JOB has initially been set to zero in the calling program.
*     .. Parameters ..
      INTEGER           NN
      PARAMETER         (NN=10)
*     .. Scalar Arguments ..
      INTEGER           IFLAG, LIWORK, LRWORK, N
*     .. Array Arguments ..
      real              RWORK(LRWORK), X(N), Y(N)
      INTEGER           IWORK(LIWORK)
*     .. Scalars in Common ..
      INTEGER           JOB
*     .. Arrays in Common ..
      real              D(NN), E(NN)
*     .. Local Scalars ..
      INTEGER           I, IFAIL
*     .. External Subroutines ..
      EXTERNAL          F04FAF
```

```
*     .. Common blocks ..
      COMMON            /USER/D, E, JOB
*     .. Executable Statements ..
      DO 20 I = 1, N
         Y(I) = X(I)
   20 CONTINUE
      IFAIL = 1
*
      CALL F04FAF(JOB,N,D,E,Y,IFAIL)
*
      IF (IFAIL.NE.0) THEN
         IFLAG = -IFAIL
      ELSE
*        After first call to F04FAF, M will be factorized, so we
*        set JOB = 1.
         IF (JOB.EQ.0) JOB = 1
      END IF
      RETURN
      END
```

9.2 Program Data

None.

9.3 Program Results

```
F04MBF Example Program Results

OUTPUT FROM F04MBF.

    N =   10          PRECON =  T     SHIFT =    0.0000000E+00
 RTOL =    1.00E-05   ITNLIM =  100

NO. OF ITERATIONS                      1
STOPPING CONDITION WAS                 1
( REQUESTED ACCURACY ACHIEVED )


REQUESTED  NORM( RBAR ) WAS                       2.22E-04
ESTIMATE OF SMALLEST ATTAINABLE NORM( RBAR )      2.46E-15

ESTIMATES OF  NORM( RBAR ),  NORM( Y )            3.47E-15    6.16E+00
                         ( Y = M**( 1/2 )* X )

ACTUAL        NORM( R ),     NORM( X )            3.72E-15    3.16E+00
ESTIMATES OF  NORM( ABAR ),  COND( ABAR )         3.60E+00    1.72E+00

Solution returned by F04MBF
    1.000    1.000    1.000    1.000    1.000
    1.000    1.000    1.000    1.000    1.000
```