NAG Fortran Library Routine Document

F04FAF

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

F04FAF calculates the approximate solution of a set of real symmetric positive-definite tridiagonal linear equations.

2 Specification

```
SUBROUTINE F04FAF(JOB, N, D, E, B, IFAIL)
INTEGER           JOB, N, IFAIL
real              D(N), E(N), B(N)
```

3 Description

F04FAF is based upon the LINPACK routine SPTSL (see Dongarra *et al.* (1979)) and solves the equations

$$Tx = b,$$

where T is a real n by n symmetric positive-definite tridiagonal matrix, using a modified symmetric Gaussian elimination algorithm to factorize T as $T = MKM^T$, where K is diagonal and M is a matrix of multipliers as described in Section 8.

When the input parameter JOB is supplied as 1, then the routine assumes that a previous call to F04FAF has already factorized T; otherwise JOB must be supplied as 0.

4 References

Dongarra J J, Moler C B, Bunch J R and Stewart G W (1979) *LINPACK Users' Guide* SIAM, Philadelphia

5 Parameters

1: JOB – INTEGER *Input*

On entry: specifies the job to be performed by F04FAF as follows:

JOB $= 0$

The matrix T is factorized and the equations $Tx = b$ are solved for x.

JOB $= 1$

The matrix T is assumed to have already been factorized by a previous call to F04FAF with JOB=0; the equations $Tx = b$ are solved for x.

2: N – INTEGER *Input*

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

Constraint: N ≥ 1.

3: D(N) – ***real*** array *Input/Output*

On entry: if JOB $= 0$, D must contain the diagonal elements of T. If JOB=1, D must contain the diagonal matrix K, as returned by a previous call of F04FAF with JOB=0.

On exit: if JOB $= 0$, D is overwritten by the diagonal matrix K of the factorization. If JOB=1, D is unchanged.

4: E(N) – ***real*** array *Input/Output*

On entry: if JOB $= 0$, E must contain the super-diagonal elements of T, stored in E(2) to E(n). If JOB=1, E must contain the off-diagonal elements of the matrix M, as returned by a previous call of F04FAF with JOB=0. E(1) is not used.

On exit: if JOB $= 0$, E(2) to E(n) are overwritten by the off-diagonal elements of the matrix M of the factorization. If JOB=1, E is unchanged.

5: B(N) – ***real*** array *Input/Output*

On entry: the right-hand side vector b.

On exit: B is overwritten by the solution vector x.

6: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

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

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, for users not familiar with this parameter the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

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, N < 1,
or JOB $\neq 0$ or 1.

IFAIL $= 2$

The matrix T is either not positive-definite or is nearly singular. This failure can only occur when JOB $= 0$ and inspection of the elements of D will give an indication of why failure has occurred. If an element of D is close to zero, then T is probably nearly singular; if an element of D is negative but not close to zero, then T is not positive-definite.

IFAIL $=$ overflow

If overflow occurs during the execution of this routine, then either T is very nearly singular or an element of the right-hand side vector b is very large. In this latter case the equations should be scaled so that no element of b is very large. Note that to preserve symmetry it is necessary to scale by a transformation of the form $(PTP^T)b = Px$, where P is a diagonal matrix.

IFAIL $=$ underflow

Any underflows that occur during the execution of this routine are harmless.

7 Accuracy

The computed factorization (see Section 8) will satisfy the equation

$$MKM^T = T + E$$

where $\|\mathrm{E}\|_p \leq 2\epsilon\|T\|_p$, $p = 1, F, \infty$,

ϵ being the ***machine precision***. The computed solution of the equations $Tx = b$, say $\bar{x}$, will satisfy an equation of the form

$$(T + F)\bar{x} = b,$$

where F can be expected to satisfy a bound of the form

$$\|F\| \leq \alpha\epsilon\|T\|,$$

α being a modest constant. This implies that the relative error in $\bar{x}$ satisfies

$$\frac{\|\bar{x} - x\|}{\|x\|} \leq c(T)\alpha\epsilon,$$

where $c(T)$ is the condition number of T with respect to inversion. Thus if T is nearly singular, $\bar{x}$ can be expected to have a large relative error.

8 Further Comments

The time taken by the routine is approximately proportional to n.

The routine eliminates the off-diagonal elements of T by simultaneously performing symmetric Gaussian elimination from the top and the bottom of T. The result is that T is factorized as

$$T = MKM^T,$$

where K is a diagonal matrix and M is a matrix of the form

$$M = \begin{pmatrix}
1 & 0 & 0 & .. & 0 & 0 & 0 & .. & 0 & 0 & 0 \\
m_2 & 1 & 0 & .. & 0 & 0 & 0 & .. & 0 & 0 & 0 \\
0 & m_3 & 1 & .. & 0 & 0 & 0 & .. & 0 & 0 & 0 \\
. & . & . & .. & . & . & . & .. & . & . & . \\
. & . & . & .. & . & . & . & .. & . & . & . \\
0 & 0 & 0 & .. & 1 & 0 & 0 & .. & 0 & 0 & 0 \\
0 & 0 & 0 & .. & m_{j+1} & 1 & m_{j+2} & .. & 0 & 0 & 0 \\
0 & 0 & 0 & .. & 0 & 0 & 1 & .. & 0 & 0 & 0 \\
. & . & . & .. & . & . & . & .. & . & . & . \\
. & . & . & .. & . & . & . & .. & . & . & . \\
0 & 0 & 0 & .. & 0 & 0 & 0 & .. & 1 & m_{n-1} & 0 \\
0 & 0 & 0 & .. & 0 & 0 & 0 & .. & 0 & 1 & m_n \\
0 & 0 & 0 & .. & 0 & 0 & 0 & .. & 0 & 0 & 1
\end{pmatrix}$$

j being the integer part of $n/2$. (For example when $n = 5, j = 2$.) The diagonal elements of K are returned in D with k_i in the ith element of D and m_i is returned in the ith element of E.

The routine fails with IFAIL=2 if any diagonal element of K is non-positive. It should be noted that T may be nearly singular even if all the diagonal elements of K are positive, but in this case at least one element of K is almost certain to be small relative to $\|T\|$. If there is any doubt as to whether or not T is nearly singular, then the user should consider examining the diagonal elements of K.

9 Example

To solve the symmetric positive-definite equations

$$Tx_1 = b_1 \quad \text{and} \quad Tx_2 = b_2$$

where

$$T = \begin{pmatrix} 4 & -2 & 0 & 0 & 0 \\ -2 & 10 & -6 & 0 & 0 \\ 0 & -6 & 29 & 15 & 0 \\ 0 & 0 & 15 & 25 & 8 \\ 0 & 0 & 0 & 8 & 5 \end{pmatrix}, \quad b_1 = \begin{pmatrix} 6 \\ 9 \\ 2 \\ 14 \\ 7 \end{pmatrix}, \quad b_2 = \begin{pmatrix} 10 \\ 4 \\ 9 \\ 65 \\ 23 \end{pmatrix}.$$

The equations are solved by two calls to F04FAF, the first with JOB = 0 and the second, using the factorization from the first call, with JOB=1.

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.

```
*     F04FAF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NMAX
      PARAMETER        (NMAX=100)
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, JOB, N
*     .. Local Arrays ..
      real             B(NMAX), D(NMAX), E(NMAX)
*     .. External Subroutines ..
      EXTERNAL         F04FAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F04FAF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      WRITE (NOUT,*)
      IF (N.LT.1 .OR. N.GT.NMAX) THEN
         WRITE (NOUT,99999) 'N is out of range: N = ', N
      ELSE
         READ (NIN,*) (D(I),I=1,N)
         IF (N.GT.1) READ (NIN,*) (E(I),I=2,N)
         READ (NIN,*) (B(I),I=1,N)
         JOB = 0
         IFAIL = 1
*
         CALL F04FAF(JOB,N,D,E,B,IFAIL)
*
         IF (IFAIL.NE.0) THEN
            WRITE (NOUT,99999) 'F04FAF fails. IFAIL =', IFAIL
         ELSE
            WRITE (NOUT,*) ' First solution vector'
            WRITE (NOUT,99998) (B(I),I=1,N)
         END IF
*
         READ (NIN,*) (B(I),I=1,N)
         JOB = 1
         IFAIL = 1
*
         CALL F04FAF(JOB,N,D,E,B,IFAIL)
*
         IF (IFAIL.NE.0) THEN
            WRITE (NOUT,99999) 'F04FAF fails. IFAIL =', IFAIL
         ELSE
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'Second solution vector'
            WRITE (NOUT,99998) (B(I),I=1,N)
         END IF
      END IF
      STOP
*
```

```
99999 FORMAT (1X,A,I5)
99998 FORMAT (1X,5F9.3)
      END
```

9.2 Program Data

```
F04FAF Example Program Data
   5
  4.0  10.0  29.0  25.0   5.0
 -2.0  -6.0  15.0   8.0
  6.0   9.0   2.0  14.0   7.0
 10.0   4.0   9.0  65.0  23.0
```

9.3 Program Results

```
 F04FAF Example Program Results

  First solution vector
     2.500    2.000    1.000   -1.000    3.000

 Second solution vector
     2.000   -1.000   -3.000    6.000   -5.000
```