F04LEF – 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

F04LEF solves a system of tridiagonal equations following the factorization by F01LEF. This routine is intended for applications such as inverse iteration as well as straightforward linear equation applications.

2 Specification

```
SUBROUTINE F04LEF(JOB, N, A, B, C, D, IN, Y, TOL, IFAIL)
INTEGER           JOB, N, IN(N), IFAIL
real              A(N), B(N), C(N), D(N), Y(N), TOL
```

3 Description

Following the factorization of the n by n tridiagonal matrix $(T - \lambda I)$ as

$$T - \lambda I = PLU$$

by F01LEF, this routine may be used to solve any of the equations

$$(T - \lambda I)x = y, \quad (T - \lambda I)^T x = y, \quad Ux = y$$

for x, the choice of equation being controlled by the parameter JOB. In each case there is an option to perturb zero or very small diagonal elements of U, this option being intended for use in applications such as inverse iteration.

4 References

[1] Wilkinson J H and Reinsch C (1971) *Handbook for Automatic Computation II, Linear Algebra* Springer-Verlag

[2] Wilkinson J H (1965) *The Algebraic Eigenvalue Problem* Oxford University Press, London

5 Parameters

1: JOB — INTEGER *Input*

On entry: JOB must specify the equations to be solved as follows:

JOB = 1

The equations $(T - \lambda I)x = y$ are to be solved, but diagonal elements of U are not to be perturbed.

JOB = −1

The equations $(T - \lambda I)x = y$ are to be solved and, if overflow would otherwise occur, diagonal elements of U are to be perturbed. See parameter TOL below.

JOB = 2

The equations $(T - \lambda I)^T x = y$ are to be solved, but diagonal elements of U are not to be perturbed.

JOB = −2

The equations $(T - \lambda I)^T x = y$ are to be solved and, if overflow would otherwise occur, diagonal elements of U are to be perturbed. See parameter TOL below.

JOB = 3

The equations $Ux = y$ are to be solved, but diagonal elements of U are not to be perturbed.

JOB = −3

The equations $Ux = y$ are to be solved and, if overflow would otherwise occur, diagonal elements of U are to be perturbed. See parameter TOL below.

2: N — INTEGER *Input*

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

Constraint: N ≥ 1.

3: A(N) — ***real*** array *Input*

On entry: the diagonal elements of U as returned by F01LEF.

4: B(N) — ***real*** array *Input*

On entry: the elements of the first super-diagonal of U as returned by F01LEF.

5: C(N) — ***real*** array *Input*

On entry: the sub-diagonal elements of L as returned by F01LEF.

6: D(N) — ***real*** array *Input*

On entry: the elements of the second super-diagonal of U as returned by F01LEF.

7: IN(N) — INTEGER array *Input*

On entry: details of the matrix P as returned by F01LEF.

8: Y(N) — ***real*** array *Input/Output*

On entry: the right-hand side vector y.

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

9: TOL — ***real*** *Input/Output*

On entry: the minimum perturbation to be made to very small diagonal elements of U. TOL is only referenced when JOB is negative. TOL should normally be chosen as about $\epsilon\|U\|$, where ϵ is the ***machine precision***, but if TOL is supplied as non-positive, then it is reset to $\epsilon \max |u_{ij}|$.

On exit: if on entry TOL is non-positive, it is reset as just described. Otherwise TOL is unchanged.

10: 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 Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

On entry, N < 1,
or JOB = 0,
or JOB < −3 or JOB > 3.

IFAIL > 1

Overflow would occur when computing the (IFAIL−1)th element of the solution vector x. This can only occur when JOB is supplied as positive and either means that a diagonal element of U is very small or that elements of the right-hand side vector y are very large.

7 Accuracy

The computed solution of the equations $(T - \lambda I)x = y$, say $\bar{x}$, will satisfy an equation of the form

$$(T - \lambda I + E)\bar{x} = y,$$

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

$$\|E\| \leq \alpha\epsilon\|T - \lambda I\|,$$

α being a modest constant and ϵ being the ***machine precision***. The computed solution of the equations $(T - \lambda I)^T x = y$ and $Ux = y$ will satisfy similar results. The above result implies that the relative error in $\bar{x}$ satisfies

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

where $c(T - \lambda I)$ is the condition number of $(T - \lambda I)$ with respect to inversion. Thus if $(T - \lambda I)$ is nearly singular, $\bar{x}$ can be expected to have a large relative error. Note that F01LEF incorporates a test for near singularity.

8 Further Comments

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

Users with single systems of tridiagonal equations to solve may wish to be aware that F04EAF requires less storage and will normally be faster than the combination of F01LEF and F04LEF, but F04EAF does not incorporate a test for near singularity.

9 Example

To solve the two sets of tridiagonal equations

$$Tx = y \text{ and } T^T x = y$$

where

$$T = \begin{pmatrix} 3.0 & 2.1 & 0 & 0 & 0 \\ 3.4 & 2.3 & -1.0 & 0 & 0 \\ 0 & 3.6 & -5.0 & 1.9 & 0 \\ 0 & 0 & 7.0 & -0.9 & 8.0 \\ 0 & 0 & 0 & -6.0 & 7.1 \end{pmatrix}, \quad y = \begin{pmatrix} 2.7 \\ -0.5 \\ 2.6 \\ 0.6 \\ 2.7 \end{pmatrix}.$$

The example program first calls F01LEF to factorize T and then two calls are made to F04LEF, one to solve $Tx = y$ and the second to solve $T^T x = y$.

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.

```
*     F04LEF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NMAX
      PARAMETER        (NMAX=50)
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
*     .. Local Scalars ..
      real             LAMBDA, TOL
      INTEGER          I, IFAIL, JOB, N
*     .. Local Arrays ..
      real             A(NMAX), B(NMAX), C(NMAX), D(NMAX), Y(NMAX),
```

```
     +                 Z(NMAX)
      INTEGER          IN(NMAX)
*     .. External Subroutines ..
      EXTERNAL         F01LEF, F04LEF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F04LEF 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,*) (A(I),I=1,N)
         IF (N.GT.1) THEN
            READ (NIN,*) (B(I),I=2,N)
            READ (NIN,*) (C(I),I=2,N)
         END IF
         TOL = 5.0e-5
         LAMBDA = 0.0e0
         IFAIL = 1
*
         CALL F01LEF(N,A,LAMBDA,B,C,TOL,D,IN,IFAIL)
*
         IF (IFAIL.NE.0) THEN
            WRITE (NOUT,*)
            WRITE (NOUT,99999) 'F01LEF fails. IFAIL =', IFAIL
         ELSE
            IF (IN(N).NE.0) THEN
               WRITE (NOUT,*) 'Matrix is singular or nearly singular'
               WRITE (NOUT,99998) 'Diagonal element', IN(N), 'is small'
            ELSE
               READ (NIN,*) (Y(I),I=1,N)
               DO 20 I = 1, N
                  Z(I) = Y(I)
   20          CONTINUE
               JOB = 1
               IFAIL = 1
*
               CALL F04LEF(JOB,N,A,B,C,D,IN,Y,TOL,IFAIL)
*
               IF (IFAIL.NE.0) THEN
                  WRITE (NOUT,*)
                  WRITE (NOUT,99999) 'F04LEF fails. IFAIL =', IFAIL
               ELSE
                  WRITE (NOUT,*)
                  WRITE (NOUT,*) 'Solution vector for T*X = Y'
                  WRITE (NOUT,99997) (Y(I),I=1,N)
               END IF
               JOB = 2
               IFAIL = 1
*
               CALL F04LEF(JOB,N,A,B,C,D,IN,Z,TOL,IFAIL)
*
               IF (IFAIL.NE.0) THEN
                  WRITE (NOUT,*)
                  WRITE (NOUT,99999) 'F04LEF fails. IFAIL =', IFAIL
               ELSE
```

```
                  WRITE (NOUT,*)
                  WRITE (NOUT,*)
     +              'Solution vector for transpose(T)*X = Y'
                  WRITE (NOUT,99997) (Z(I),I=1,N)
               END IF
            END IF
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,A,I5)
99998 FORMAT (1X,A,I4,A)
99997 FORMAT (1X,5F9.3)
      END
```

9.2 Program Data

```
F04LEF Example Program Data
   5
  3.0   2.3  -5.0  -0.9   7.1
  2.1  -1.0   1.9   8.0
  3.4   3.6   7.0  -6.0
  2.7  -0.5   2.6   0.6   2.7
```

9.3 Program Results

```
 F04LEF Example Program Results


 Solution vector for T*X = Y
    -4.000    7.000    3.000   -4.000   -3.000

 Solution vector for transpose(T)*X = Y
    -4.630    4.880   -0.555    0.672   -0.377
```