NAG Library Routine Document

F04LEF

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, IPIV, Y, TOL, IFAIL)
INTEGER            JOB, N, IPIV(N), IFAIL
REAL (KIND=nag_wp) 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, F04LEF may be used to solve any of the equations

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

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

Wilkinson J H (1965) *The Algebraic Eigenvalue Problem* Oxford University Press, Oxford

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

5 Parameters

1: JOB – INTEGER *Input*

On entry: must specify the equations to be solved.

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.

JOB $= 2$
: The equations $(T - \lambda I)^{\mathrm{T}}x = y$ are to be solved, but diagonal elements of U are not to be perturbed.

JOB $= -2$
: The equations $(T - \lambda I)^{\mathrm{T}}x = y$ are to be solved and, if overflow would otherwise occur, diagonal elements of U are to be perturbed. See parameter TOL.

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.

2: N – INTEGER *Input*

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

Constraint: N ≥ 1.

3: A(N) – REAL (KIND=nag_wp) array *Input*

On entry: the diagonal elements of U as returned by F04LEF.

4: B(N) – REAL (KIND=nag_wp) array *Input*

On entry: the elements of the first superdiagonal of U as returned by F04LEF.

5: C(N) – REAL (KIND=nag_wp) array *Input*

On entry: the subdiagonal elements of L as returned by F04LEF.

6: D(N) – REAL (KIND=nag_wp) array *Input*

On entry: the elements of the second superdiagonal of U as returned by F04LEF.

7: IPIV(N) – INTEGER array *Input*

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

8: Y(N) – REAL (KIND=nag_wp) array *Input/Output*

On entry: the right-hand side vector y.

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

9: TOL – REAL (KIND=nag_wp) *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 $\epsilon \max\left|u_{ij}\right|$.

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. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction for details.

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, if you are 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.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

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 $= 0$,
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.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.8 in the Essential Introduction for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.7 in the Essential Introduction for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

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 $(T - \lambda I)^{\mathrm{T}}x = y$ 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 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken by F04LEF is approximately proportional to n.

If you have single systems of tridiagonal equations to solve you are advised that F07CAF (DGTSV) requires less storage and will normally be faster than the combination of F01LEF and F04LEF, but F07CAF (DGTSV) does not incorporate a test for near singularity.

10 Example

This example solves the two sets of tridiagonal equations

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

where

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

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 $T^{\mathrm{T}}x = y$.

10.1 Program Text

```
    Program f04lefe

!     F04LEF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: f01lef, f04lef, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: lambda, tol
      Integer                          :: ifail, job, n
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:), b(:), c(:), d(:), y(:), z(:)
      Integer, Allocatable             :: ipiv(:)
!     .. Executable Statements ..
      Write (nout,*) 'F04LEF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) n
      Allocate (a(n),b(n),c(n),d(n),y(n),z(n),ipiv(n))
      Read (nin,*) a(1:n)
      If (n>1) Then
        Read (nin,*) b(2:n)
        Read (nin,*) c(2:n)
      End If
      tol = 5.0E-5_nag_wp
      lambda = 0.0E0_nag_wp

!     ifail: behaviour on error exit
!             =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
      Call f01lef(n,a,lambda,b,c,tol,d,ipiv,ifail)

      If (ipiv(n)/=0) Then
        Write (nout,*) 'Matrix is singular or nearly singular'
        Write (nout,99999) 'Diagonal element', ipiv(n), 'is small'
      Else
        Read (nin,*) y(1:n)
        z(1:n) = y(1:n)
        job = 1
```

```
        ifail = 0
        Call f04lef(job,n,a,b,c,d,ipiv,y,tol,ifail)

        Write (nout,*)
        Write (nout,*) 'Solution vector for T*X = Y'
        Write (nout,99998) y(1:n)
        job = 2

        ifail = 0
        Call f04lef(job,n,a,b,c,d,ipiv,z,tol,ifail)

        Write (nout,*)
        Write (nout,*) 'Solution vector for transpose(T)*X = Y'
        Write (nout,99998) z(1:n)
      End If

99999 Format (1X,A,I4,A)
99998 Format (1X,5F9.3)
    End Program f04lefe
```

10.2 Program Data

```
F04LEF Example Program Data
   5                              : n
  3.0   2.3  -5.0  -0.9   7.1    : vector A
  2.1  -1.0   1.9   8.0          : vector B
  3.4   3.6   7.0  -6.0          : vector C
  2.7  -0.5   2.6   0.6   2.7    : vector Y
```

10.3 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
```