NAG Library Routine Document

F04MFF

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

F04MFF updates the solution of the equations $Tx = b$, where T is a real symmetric positive definite Toeplitz matrix.

2 Specification

```
SUBROUTINE F04MFF (N, T, B, X, P, WORK, IFAIL)

INTEGER             N, IFAIL
REAL (KIND=nag_wp) T(0:*), B(*), X(*), P, WORK(*)
```

3 Description

F04MFF solves the equations

$$T_n x_n = b_n,$$

where T_n is the n by n symmetric positive definite Toeplitz matrix

$$T_n = \begin{pmatrix} \tau_0 & \tau_1 & \tau_2 & \dots & \tau_{n-1} \\ \tau_1 & \tau_0 & \tau_1 & \dots & \tau_{n-2} \\ \tau_2 & \tau_1 & \tau_0 & \dots & \tau_{n-3} \\ . & . & . & & . \\ \tau_{n-1} & \tau_{n-2} & \tau_{n-3} & \dots & \tau_0 \end{pmatrix}$$

and b_n is the n-element vector $b_n = (\beta_1 \beta_2 \dots \beta_n)^{\mathrm{T}}$, given the solution of the equations

$$T_{n-1} x_{n-1} = b_{n-1}.$$

This routine will normally be used to successively solve the equations

$$T_k x_k = b_k, \quad k = 1, 2, \dots, n.$$

If it is desired to solve the equations for a single value of n, then routine F04FFF may be called. This routine uses the method of Levinson (see Levinson (1947) and Golub and Van Loan (1996)).

4 References

Bunch J R (1985) Stability of methods for solving Toeplitz systems of equations *SIAM J. Sci. Statist. Comput.* **6** 349–364

Bunch J R (1987) The weak and strong stability of algorithms in numerical linear algebra *Linear Algebra Appl.* **88/89** 49–66

Cybenko G (1980) The numerical stability of the Levinson–Durbin algorithm for Toeplitz systems of equations *SIAM J. Sci. Statist. Comput.* **1** 303–319

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

Levinson N (1947) The Weiner RMS error criterion in filter design and prediction *J. Math. Phys.* **25** 261–278

5 Parameters

1: N – INTEGER *Input*

On entry: the order of the Toeplitz matrix T.

Constraint: $\mathrm{N} \geq 0$. When $\mathrm{N} = 0$, then an immediate return is effected.

2: T(0 : *) – REAL (KIND=nag_wp) array *Input*

Note: the dimension of the array T must be at least $\max(1, \mathrm{N})$.

On entry: $\mathrm{T}(i)$ must contain the value τ_i, for $i = 0, 1, \ldots, \mathrm{N} - 1$.

Constraint: $\mathrm{T}(0) > 0.0$. Note that if this is not true, then the Toeplitz matrix cannot be positive definite.

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

Note: the dimension of the array B must be at least $\max(1, \mathrm{N})$.

On entry: the right-hand side vector b_n.

4: X(*) – REAL (KIND=nag_wp) array *Input/Output*

Note: the dimension of the array X must be at least $\max(1, \mathrm{N})$.

On entry: with $\mathrm{N} > 1$ the $(n - 1)$ elements of the solution vector x_{n-1} as returned by a previous call to F04MFF. The element X(N) need not be specified.

On exit: the solution vector x_n.

5: P – REAL (KIND=nag_wp) *Output*

On exit: the reflection coefficient p_{n-1}. (See Section 9.)

6: WORK(*) – REAL (KIND=nag_wp) array *Input/Output*

Note: the dimension of the array WORK must be at least $\max(1, 2 \times \mathrm{N} - 1)$.

On entry: with $\mathrm{N} > 2$ the elements of WORK should be as returned from a previous call to F04MFF with $(\mathrm{N} - 1)$ as the parameter N.

On exit: the first $(\mathrm{N} - 1)$ elements of WORK contain the solution to the Yule–Walker equations

$$T_{n-1}y_{n-1} = -t_{n-1},$$

where $t_{n-1} = (\tau_1\tau_2 \ldots \tau_{n-1})^{\mathrm{T}}$.

7: 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: $\mathrm{IFAIL} = 0$ 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 < 0,
or T(0) ≤ 0.0.

IFAIL $= 1$

The Toeplitz matrix T_n is not positive definite to working accuracy. If, on exit, P is close to unity, then T_n was probably close to being singular.

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 certainly satisfies

$$r = T_n x_n - b_n,$$

where $\|r\|_1$ is approximately bounded by

$$\|r\|_1 \leq c\epsilon C(T_n),$$

c being a modest function of n, ϵ being the ***machine precision*** and $C(T)$ being the condition number of T with respect to inversion. This bound is almost certainly pessimistic, but it seems unlikely that the method of Levinson is backward stable, so caution should be exercised when T_n is ill-conditioned. The following bound on T_n^{-1} holds:

$$\max\left(\frac{1}{\prod_{i=1}^{n-1}\left(1-p_i^2\right)}, \frac{1}{\prod_{i=1}^{n-1}\left(1-p_i\right)}\right) \leq \left\|T_n^{-1}\right\|_1 \leq \prod_{i=1}^{n-1}\left(\frac{1+|p_i|}{1-|p_i|}\right).$$

(See Golub and Van Loan (1996).) The norm of T_n^{-1} may also be estimated using routine F04YDF. For further information on stability issues see Bunch (1985), Bunch (1987), Cybenko (1980) and Golub and Van Loan (1996).

8 Parallelism and Performance

F04MFF is not threaded by NAG in any implementation.

F04MFF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

The number of floating-point operations used by this routine is approximately $8n$.

If y_i is the solution of the equations

$$T_i y_i = -(\tau_1 \tau_2 \ldots \tau_i)^{\mathrm{T}},$$

then the reflection coefficient p_i is defined as the ith element of y_i.

10 Example

This example finds the solution of the equations $T_k x_k = b_k$, $k = 1, 2, 3, 4$, where

$$T_4 = \begin{pmatrix} 4 & 3 & 2 & 1 \\ 3 & 4 & 3 & 2 \\ 2 & 3 & 4 & 3 \\ 1 & 2 & 3 & 4 \end{pmatrix} \quad \text{and} \quad b_4 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

10.1 Program Text

```
    Program f04mffe

!     F04MFF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: f04mff, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: p
      Integer                          :: ifail, k, n
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: b(:), t(:), work(:), x(:)
!     .. Executable Statements ..
      Write (nout,*) 'F04MFF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) n
      Allocate (b(n),t(0:n-1),work(2*n-1),x(n))
      Read (nin,*) t(0:n-1)
      Read (nin,*) b(1:n)
      Do k = 1, n
!       ifail: behaviour on error exit
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        ifail = 0
        Call f04mff(k,t,b,x,p,work,ifail)

        Write (nout,*)
        Write (nout,99999) 'Solution for system of order', k
        Write (nout,99998) x(1:k)
        If (k>1) Then
          Write (nout,*) 'Reflection coefficient'
          Write (nout,99998) p
```

```
        End If
      End Do

99999 Format (1X,A,I5)
99998 Format (1X,5F9.4)
    End Program f04mffe
```

10.2 Program Data

```
F04MFF Example Program Data

   4                      : n
   4.0  3.0  2.0  1.0     : vector T
   1.0  1.0  1.0  1.0     : vector B
```

10.3 Program Results

```
 F04MFF Example Program Results


 Solution for system of order    1
    0.2500

 Solution for system of order    2
    0.1429   0.1429
 Reflection coefficient
   -0.7500

 Solution for system of order    3
    0.1667   0.0000   0.1667
 Reflection coefficient
    0.1429

 Solution for system of order    4
    0.2000   0.0000  -0.0000   0.2000
 Reflection coefficient
    0.1667
```