# NAG Library Routine Document

# F04FFF

**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

F04FFF solves the equations $Tx = b$, where $T$ is a real symmetric positive definite Toeplitz matrix.

## 2 Specification

```
SUBROUTINE F04FFF (N, T, B, X, WANTP, P, WORK, IFAIL)
INTEGER             N, IFAIL
REAL (KIND=nag_wp) T(0:*), B(*), X(N), P(*), WORK(2*(N-1))
LOGICAL             WANTP
```

## 3 Description

F04FFF solves the equations

$$Tx = b,$$

where $T$ is the $n$ by $n$ symmetric positive definite Toeplitz matrix

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

and $b$ is an $n$-element vector.

The routine uses the method of Levinson (see Levinson (1947) and Golub and Van Loan (1996)). Optionally, the reflection coefficients for each step may also be returned.

## 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 Arguments

1: N – INTEGER *Input*

*On entry*: the order of the Toeplitz matrix $T$.

*Constraint*: N $\geq$ 0. When 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*: T$(i)$ must contain the value $\tau_i$, for $i = 0, 1, \ldots, \mathrm{N}-1$.

*Constraint*: 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 $b$.

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

*On exit*: the solution vector $x$.

5: WANTP – LOGICAL *Input*

*On entry*: must be set to .TRUE. if the reflection coefficients are required, and must be set to .FALSE. otherwise.

6: P$(*)$ – REAL (KIND=nag_wp) array *Output*

**Note**: the dimension of the array P must be at least $\max(1, \mathrm{N}-1)$ if WANTP = .TRUE., and at least 1 otherwise.

*On exit*: with WANTP as .TRUE., the $i$th element of P contains the reflection coefficient, $p_i$, for the $i$th step, for $i = 1, 2, \ldots, \mathrm{N}-1$. (See Section 9.) If WANTP is .FALSE., then P is not referenced.

7: WORK$(2 \times (\mathrm{N}-1))$ – REAL (KIND=nag_wp) array *Workspace*

8: IFAIL – INTEGER *Input/Output*

*On entry*: IFAIL must be set to $0$, $-1$ or $1$. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation 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, because for this routine the values of the output arguments may be useful even if IFAIL $\neq 0$ on exit, the recommended value is $-1$. **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).

**Note**: F04FFF may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the routine:

IFAIL $= -1$

On entry, N $< 0$,
or T$(0) \leq 0.0$.

IFAIL $> 0$

The principal minor of order IFAIL of the Toeplitz matrix is not positive definite to working accuracy. The first (IFAIL $-$ 1) elements of X return the solution of the equations

$$T_{\mathrm{IFAIL}-1}x = (b_1, b_2, \ldots, b_{\mathrm{IFAIL}-1})^{\mathrm{T}},$$

where $T_k$ is the $k$th principal minor of $T$.

IFAIL $= -99$

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

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

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

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

## 7 Accuracy

The computed solution of the equations certainly satisfies

$$r = Tx - b,$$

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

$$\|r\| \leq c\epsilon C(T),$$

$c$ being a modest function of $n$, $\epsilon$ 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 $T$ is ill-conditioned. The following bound on $T^{-1}$ holds:

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

(See Golub and Van Loan (1996).) The norm of $T^{-1}$ 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

F04FFF 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 F04FFF is approximately $4n^2$.

If $y_i$ is the solution of the equations

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

then the partial correlation coefficient $p_i$ is defined as the $i$th element of $y_i$.

## 10 Example

This example finds the solution of the equations $Tx = b$, where

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

### 10.1 Program Text

```
    Program f04fffe

!     F04FFF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: f04fff, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: ifail, n
      Logical                          :: wantp
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: b(:), p(:), t(:), work(:), x(:)
!     .. Executable Statements ..
      Write (nout,*) 'F04FFF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) n
      Write (nout,*)
      Allocate (b(n),p(n-1),t(0:n-1),work(2*(n-1)),x(n))
      Read (nin,*) t(0:n-1)
      Read (nin,*) b(1:n)
      wantp = .True.

!     ifail: behaviour on error exit
!             =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 1
      Call f04fff(n,t,b,x,wantp,p,work,ifail)

      If (ifail==0) Then
        Write (nout,*)
        Write (nout,*) 'Solution vector'
        Write (nout,99998) x(1:n)
        If (wantp) Then
          Write (nout,*)
          Write (nout,*) 'Reflection coefficients'
          Write (nout,99998) p(1:n-1)
        End If
      Else If (ifail>0) Then
        Write (nout,*)
        Write (nout,99999) 'Solution for system of order', ifail - 1
        Write (nout,99998) x(1:ifail-1)
        If (wantp) Then
```

```
          Write (nout,*)
          Write (nout,*) 'Reflection coefficients'
          Write (nout,99998) p(1:ifail-1)
        End If
      Else
        Write (nout,99997) ifail
      End If

99999 Format (1X,A,I5)
99998 Format (1X,5F9.4)
99997 Format (1X,' ** F04FFF returned with IFAIL = ',I5)
    End Program f04fffe
```

## 10.2 Program Data

```
F04FFF 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

```
 F04FFF Example Program Results


 Solution vector
    0.2000   0.0000  -0.0000   0.2000

 Reflection coefficients
   -0.7500   0.1429   0.1667
```