# NAG Library Routine Document

# F01EFF

**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

F01EFF computes the matrix function, $f(A)$, of a real symmetric $n$ by $n$ matrix $A$. $f(A)$ must also be a real symmetric matrix.

## 2 Specification

```
SUBROUTINE F01EFF (UPLO, N, A, LDA, F, IUSER, RUSER, IFLAG, IFAIL)

INTEGER             N, LDA, IUSER(*), IFLAG, IFAIL
REAL (KIND=nag_wp) A(LDA,*), RUSER(*)
CHARACTER(1)        UPLO
EXTERNAL            F
```

## 3 Description

$f(A)$ is computed using a spectral factorization of $A$

$$A = QDQ^{\mathrm{T}},$$

where $D$ is the diagonal matrix whose diagonal elements, $d_i$, are the eigenvalues of $A$, and $Q$ is an orthogonal matrix whose columns are the eigenvectors of $A$. $f(A)$ is then given by

$$f(A) = Qf(D)Q^{\mathrm{T}},$$

where $f(D)$ is the diagonal matrix whose $i$th diagonal element is $f(d_i)$. See for example Section 4.5 of Higham (2008). $f(d_i)$ is assumed to be real.

## 4 References

Higham N J (2008) *Functions of Matrices: Theory and Computation* SIAM, Philadelphia, PA, USA

## 5 Parameters

1: UPLO – CHARACTER(1) *Input*

*On entry*: if UPLO = 'U', the upper triangle of the matrix $A$ is stored.

If UPLO = 'L', the lower triangle of the matrix $A$ is stored.

*Constraint*: UPLO = 'U' or 'L'.

2: N – INTEGER *Input*

*On entry*: $n$, the order of the matrix $A$.

*Constraint*: N $\geq$ 0.

3: A(LDA,$*$) – REAL (KIND=nag_wp) array *Input/Output*

**Note**: the second dimension of the array A must be at least N.

*On entry*: the $n$ by $n$ symmetric matrix $A$.

If UPLO = 'U', the upper triangular part of $A$ must be stored and the elements of the array below the diagonal are not referenced.

If UPLO = 'L', the lower triangular part of $A$ must be stored and the elements of the array above the diagonal are not referenced.

*On exit*: if IFAIL $= 0$, the upper or lower triangular part of the $n$ by $n$ matrix function, $f(A)$.

4: LDA – INTEGER *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F01EFF is called.

*Constraint*: LDA $\geq \max(1, \mathrm{N})$.

5: F – SUBROUTINE, supplied by the user. *External Procedure*

The subroutine F evaluates $f(z_i)$ at a number of points $z_i$.

The specification of F is:

```
SUBROUTINE F (IFLAG, N, X, FX, IUSER, RUSER)

INTEGER              IFLAG, N, IUSER(*)
REAL (KIND=nag_wp) X(N), FX(N), RUSER(*)
```

1: IFLAG – INTEGER *Input/Output*

*On entry*: IFLAG will be zero.

*On exit*: IFLAG should either be unchanged from its entry value of zero, or may be set nonzero to indicate that there is a problem in evaluating the function $f(x)$; for instance $f(x)$ may not be defined, or may be complex. If IFLAG is returned as nonzero then F01EFF will terminate the computation, with IFAIL $= -6$.

2: N – INTEGER *Input*

*On entry*: $n$, the number of function values required.

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

*On entry*: the $n$ points $x_1, x_2, \ldots, x_n$ at which the function $f$ is to be evaluated.

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

*On exit*: the $n$ function values. FX$(i)$ should return the value $f(x_i)$, for $i = 1, 2, \ldots, n$.

5: IUSER$(*)$ – INTEGER array *User Workspace*
6: RUSER$(*)$ – REAL (KIND=nag_wp) array *User Workspace*

F is called with the parameters IUSER and RUSER as supplied to F01EFF. You are free to use the arrays IUSER and RUSER to supply information to F as an alternative to using COMMON global variables.

F must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which F01EFF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

6: IUSER$(*)$ – INTEGER array *User Workspace*
7: RUSER$(*)$ – REAL (KIND=nag_wp) array *User Workspace*

IUSER and RUSER are not used by F01EFF, but are passed directly to F and may be used to pass information to this routine as an alternative to using COMMON global variables.

8: IFLAG – INTEGER *Output*

*On exit*: IFLAG $= 0$, unless you have set IFLAG nonzero inside F, in which case IFLAG will be the value you set and IFAIL will be set to IFAIL $= -6$.

9: 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 $< 0$ and IFAIL $\neq -999$ or $-6$

If IFAIL $= -i$, the $i$th argument had an illegal value.

IFAIL $= -6$

IFLAG has been set nonzero by the user.

IFAIL $= -999$

Internal memory allocation failed.

The integer allocatable memory required is N, and the real allocatable memory required is approximately $(\mathrm{N} + nb + 4) \times \mathrm{N}$, where $nb$ is the block size required by F08FAF (DSYEV).

IFAIL $= i$ and IFAIL $> 0$

The algorithm to compute the spectral factorization failed to converge; $i$ off-diagonal elements of an intermediate tridiagonal form did not converge to zero (see F08FAF (DSYEV)).

**Note:** this failure is unlikely to occur.

# 7 Accuracy

Provided that $f(D)$ can be computed accurately then the computed matrix function will be close to the exact matrix function. See Section 10.2 of Higham (2008) for details and further discussion.

# 8 Further Comments

The cost of the algorithm is $O(n^3)$ plus the cost of evaluating $f(D)$. If $\hat{\lambda}_i$ is the $i$th computed eigenvalue of $A$, then the user-supplied subroutine F will be asked to evaluate the function $f$ at $f\left(\hat{\lambda}_i\right)$, $i = 1, 2, \ldots, n$.

For further information on matrix functions, see Higham (2008).

F01FFF can be used to find the matrix function $f(A)$ for a complex Hermitian matrix $A$.

# 9 Example

This example finds the matrix cosine, $\cos(A)$, of the symmetric matrix

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 2 & 3 \\ 3 & 2 & 1 & 2 \\ 4 & 3 & 2 & 1 \end{pmatrix}.$$

## 9.1 Program Text

```
!   F01EFF Example Program Text
!   Mark 24 Release. NAG Copyright 2012.

    Module f01effe_mod

!     F01EFF Example Program Module:
!            Parameters and User-defined Routines

!     nin:       the input channel number
!     nout:      the output channel number

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter                   :: nin = 5, nout = 6
    Contains
      Subroutine f(iflag,n,x,fx,iuser,ruser)

!       .. Scalar Arguments ..
        Integer, Intent (Inout)              :: iflag
        Integer, Intent (In)                 :: n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out)     :: fx(n)
        Real (Kind=nag_wp), Intent (Inout)   :: ruser(*)
        Real (Kind=nag_wp), Intent (In)      :: x(n)
        Integer, Intent (Inout)              :: iuser(*)
!       .. Intrinsic Procedures ..
        Intrinsic                            :: cos
!       .. Executable Statements ..
        fx(1:n) = cos(x(1:n))

        Return
      End Subroutine f
    End Module f01effe_mod
    Program f01effe

!     F01EFF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: f01eff, nag_wp, x04caf
      Use f01effe_mod, Only: f, nin, nout
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Integer                              :: i, ierr, ifail, iflag, lda, n
      Character (1)                        :: uplo
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable      :: a(:,:)
      Real (Kind=nag_wp)                   :: ruser(1)
      Integer                              :: iuser(1)
!     .. Executable Statements ..
      Write (nout,*) 'F01EFF Example Program Results'
      Write (nout,*)
      Flush (nout)
!     Skip heading in data file
      Read (nin,*)
```

```
      Read (nin,*) n
      Read (nin,*) uplo

      lda = n
      Allocate (a(lda,n))

!     Read A from data file

      If (uplo=='U' .Or. uplo=='u') Then
        Read (nin,*)(a(i,i:n),i=1,n)
      Else
        Read (nin,*)(a(i,1:i),i=1,n)
      End If

!     Find f( A )

      ifail = 0
      Call f01eff(uplo,n,a,lda,f,iuser,ruser,iflag,ifail)

!     Print solution

      ierr = 0
      Call x04caf(uplo,'N',n,n,a,lda,'Symmetric f(A)',ierr)

    End Program f01effe
```

## 9.2 Program Data

```
F01EFF Example Program Data

  4                     :Value of N
 'U'                    :Value of UPLO

  1.0   2.0   3.0   4.0
        1.0   2.0   3.0
              1.0   2.0
                    1.0  :End of matrix A
```

## 9.3 Program Results

```
 F01EFF Example Program Results

 Symmetric f(A)
          1       2       3       4
 1  -0.5420 -0.6612 -0.0261  0.1580
 2           0.2306 -0.3396 -0.0261
 3                   0.2306 -0.6612
 4                          -0.5420
```