NAG Library Routine Document

F01FFF

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

F01FFF computes the matrix function, $f(A)$, of a complex Hermitian n by n matrix A. $f(A)$ must also be a complex Hermitian matrix.

2 Specification

```
SUBROUTINE F01FFF (UPLO, N, A, LDA, F, IUSER, RUSER, IFLAG, IFAIL)
INTEGER                 N, LDA, IUSER(*), IFLAG, IFAIL
REAL (KIND=nag_wp)      RUSER(*)
COMPLEX (KIND=nag_wp) A(LDA,*)
CHARACTER(1)            UPLO
EXTERNAL                F
```

3 Description

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

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

where D is the real diagonal matrix whose diagonal elements, d_i, are the eigenvalues of A, Q is a unitary matrix whose columns are the eigenvectors of A and Q^{H} is the conjugate transpose of Q. $f(A)$ is then given by

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

where $f(D)$ is the diagonal matrix whose ith 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 Arguments

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, $*$) – COMPLEX (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 Hermitian 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 F01FFF 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 F01FFF 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 arguments IUSER and RUSER as supplied to F01FFF. You should use the arrays IUSER and RUSER to supply information to F.

F must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub) program from which F01FFF is called. Arguments 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 F01FFF, but are passed directly to F and should be used to pass information to this routine.

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 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, if you are not familiar with this argument, 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

The computation of the spectral factorization failed to converge.

IFAIL $= -1$

On entry, UPLO $= \langle value \rangle$.
Constraint: UPLO = 'L' or 'U'.

IFAIL $= -2$

On entry, N $= \langle value \rangle$.
Constraint: N ≥ 0.

IFAIL $= -3$

An internal error occurred when computing the spectral factorization. Please contact NAG.

IFAIL $= -4$

On entry, LDA $= \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: LDA $\geq$ N.

IFAIL $= -6$

IFLAG was set to a nonzero value in F.

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

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 Parallelism and Performance

F01FFF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F01FFF 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 integer allocatable memory required is N, the real allocatable memory required is $4 \times \mathrm{N} - 2$ and the complex allocatable memory required is approximately $(\mathrm{N} + nb + 1) \times \mathrm{N}$, where nb is the block size required by F08FNF (ZHEEV).

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

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

F01EFF can be used to find the matrix function $f(A)$ for a real symmetric matrix A.

10 Example

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

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

10.1 Program Text

```
!   F01FFF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.

    Module f01fffe_mod

!     F01FFF 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
!     .. Accessibility Statements ..
      Private
      Public                           :: f
!     .. Parameters ..
      Integer, Parameter, Public       :: 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 f01fffe_mod
    Program f01fffe

!     F01FFF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: f01fff, nag_wp, x04daf
      Use f01fffe_mod, Only: f, nin, nout
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Integer                          :: i, ierr, ifail, iflag, lda, n
      Character (1)                    :: uplo
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:,:)
      Real (Kind=nag_wp)               :: ruser(1)
      Integer                          :: iuser(1)
!     .. Executable Statements ..
      Write (nout,*) 'F01FFF 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 f01fff(uplo,n,a,lda,f,iuser,ruser,iflag,ifail)

!     Print solution
```

```
      ierr = 0
      Call x04daf(uplo,'N',n,n,a,lda,'Hermitian f(A)',ierr)

    End Program f01fffe
```

10.2 Program Data

```
F01FFF Example Program Data

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

  (1.0, 0.0)  (2.0, 1.0)  (3.0, 2.0)  (4.0, 3.0)
              (1.0, 0.0)  (2.0, 1.0)  (3.0, 2.0)
                          (1.0, 0.0)  (2.0, 1.0)
                                      (1.0, 0.0)  :End of matrix A
```

10.3 Program Results

```
 F01FFF Example Program Results

 Hermitian f(A)
          1       2       3       4
 1   0.0904 -0.3377 -0.1009 -0.1092
     0.0000 -0.0273 -0.0594 -0.1586

 2           0.4265 -0.3139 -0.1009
             0.0000 -0.0273 -0.0594

 3                   0.4265 -0.3377
                     0.0000 -0.0273

 4                           0.0904
                             0.0000
```