NAG Library Routine Document

F01ELF

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

F01ELF computes the matrix function, $f(A)$, of a real n by n matrix A. Numerical differentiation is used to evaluate the derivatives of f when they are required.

2 Specification

```
SUBROUTINE F01ELF (N, A, LDA, F, IUSER, RUSER, IFLAG, IMNORM, IFAIL)

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

3 Description

$f(A)$ is computed using the Schur–Parlett algorithm described in Higham (2008) and Davies and Higham (2003). The coefficients of the Taylor series used in the algorithm are evaluated using the numerical differentiation algorithm of Lyness and Moler (1967).

The scalar function f is supplied via subroutine F which evaluates $f(z_i)$ at a number of points z_i.

4 References

Davies P I and Higham N J (2003) A Schur–Parlett algorithm for computing matrix functions. *SIAM J. Matrix Anal. Appl.* **25(2)** 464–485

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

Lyness J N and Moler C B (1967) Numerical differentiation of analytic functions *SIAM J. Numer. Anal.* **4(2)** 202–210

5 Parameters

1: N – INTEGER *Input*

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

Constraint: $\mathrm{N} \geq 0$.

2: 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 matrix A.

On exit: the n by n matrix, $f(A)$.

3: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F01ELF is called.

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

4: 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, NZ, Z, FZ, IUSER, RUSER)

INTEGER                 IFLAG, NZ, IUSER(*)
REAL (KIND=nag_wp)      RUSER(*)
COMPLEX (KIND=nag_wp) Z(NZ), FZ(NZ)
```

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 $f(z_i)$; for instance $f(z_i)$ may not be defined. If IFLAG is returned as nonzero then F01ELF will terminate the computation, with IFAIL $= 2$.

2: NZ – INTEGER *Input*

On entry: n_z, the number of function values required.

3: Z(NZ) – COMPLEX (KIND=nag_wp) array *Input*

On entry: the n_z points $z_1, z_2, \dots, z_{n_z}$ at which the function f is to be evaluated.

4: FZ(NZ) – COMPLEX (KIND=nag_wp) array *Output*

On exit: the n_z function values. FZ(i) should return the value $f(z_i)$, for $i = 1, 2, \dots, n_z$. If z_i lies on the real line, then so must $f(z_i)$.

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 F01ELF. 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 F01ELF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

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

IUSER and RUSER are not used by F01ELF, but are passed directly to F and may be used to pass information to this routine as an alternative to using COMMON global variables.

7: IFLAG – INTEGER *Output*

On exit: IFLAG $= 0$, unless IFLAG has been set nonzero inside F, in which case IFLAG will be the value set and IFAIL will be set to IFAIL $= 2$.

8: IMNORM – REAL (KIND=nag_wp) *Output*

On exit: if A has complex eigenvalues, F01ELF will use complex arithmetic to compute $f(A)$. The imaginary part is discarded at the end of the computation, because it will theoretically vanish. IMNORM contains the 1-norm of the imaginary part, which should be used to check that the routine has given a reliable answer.

If A has real eigenvalues, F01ELF uses real arithmetic and IMNORM $= 0$.

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 $= 1$

A Taylor series failed to converge after 40 terms. Further Taylor series coefficients can no longer reliably be obtained by numerical differentiation.

IFAIL $= 2$

IFLAG has been set nonzero by the user.

IFAIL $= 3$

There was an error whilst reordering the Schur form of A.
Note: this failure should not occur and suggests that the routine has been called incorrectly.

IFAIL $= 4$

The routine was unable to compute the Schur decomposition of A.
Note: this failure should not occur and suggests that the routine has been called incorrectly.

IFAIL $= 5$

An unexpected internal error occurred. Please contact NAG.

IFAIL $= -1$

Input argument number $\langle$*value*$\rangle$ is invalid.

IFAIL $= -3$

On entry, parameter LDA is invalid.
Constraint: LDA $\geq$ N.

IFAIL $= -999$

Allocation of memory failed.

7 Accuracy

For a normal matrix A (for which $A^{\mathrm{T}}A = AA^{\mathrm{T}}$) the Schur decomposition is diagonal and the algorithm reduces to evaluating f at the eigenvalues of A and then constructing $f(A)$ using the Schur vectors. See Section 9.4 of Higham (2008) for further discussion of the Schur–Parlett algorithm, and Lyness and Moler (1967) for discussion of the numerical differentiation subroutine.

8 Further Comments

The integer allocatable memory required is n. If A has real eigenvalues then up to $6n^2$ of real allocatable memory may be required. If A has complex eigenvalues then up to $6n^2$ of complex allocatable memory may be required.

The cost of the Schur–Parlett algorithm depends on the spectrum of A, but is roughly between $28n^3$ and $n^4/3$ floating point operations. There is an additional cost in numerically differentiating f, in order to obtain the Taylor series coefficients. If the derivatives of f are known analytically, then F01EMF can be used to evaluate $f(A)$ more accurately. If A is real symmetric then it is recommended that F01EFF be used as it is more efficient and, in general, more accurate than F01ELF.

For any z on the real line, $f(z)$ must be real. f must also be complex analytic ont he spectrum of A. These conditions ensure that $f(A)$ is real for real A.

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

If estimates of the condition number of the matrix function are required then F01JBF should be used.

F01FLF can be used to find the matrix function $f(A)$ for a complex matrix A.

9 Example

This example finds $\cos 2A$ where

$$A = \begin{pmatrix} 3 & 0 & 1 & 2 \\ -1 & 1 & 3 & 1 \\ 0 & 2 & 2 & 1 \\ 2 & 1 & -1 & 1 \end{pmatrix}.$$

9.1 Program Text

```
!   F01ELF Example Program Text

!   Mark 24 Release. NAG Copyright 2012.

    Module f01elfe_mod

!     F01ELF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
    Contains
      Subroutine fcos2(iflag,nz,z,fz,iuser,ruser)

!       .. Use Statements ..
        Use nag_library, Only: nag_wp
!       .. Scalar Arguments ..
        Integer, Intent (Inout)             :: iflag
        Integer, Intent (In)                :: nz
!       .. Array Arguments ..
        Complex (Kind=nag_wp), Intent (Out)  :: fz(nz)
        Complex (Kind=nag_wp), Intent (In)   :: z(nz)
        Real (Kind=nag_wp), Intent (Inout)   :: ruser(*)
        Integer, Intent (Inout)             :: iuser(*)
!       .. Intrinsic Procedures ..
        Intrinsic                           :: cos
!       .. Executable Statements ..
        Continue
        fz(1:nz) = cos((2.0E0_nag_wp,0.0E0_nag_wp)*z(1:nz))
        Return

      End Subroutine fcos2
```

```
    End Module f01elfe_mod

    Program f01elfe

!     F01ELF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: f01elf, nag_wp, x04caf
      Use f01elfe_mod, Only: fcos2
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter                   :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)                   :: imnorm
      Integer                              :: i, ifail, iflag, lda, n
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable      :: a(:,:)
      Real (Kind=nag_wp)                   :: ruser(1)
      Integer                              :: iuser(1)
!     .. Executable Statements ..
      Write (nout,*) 'F01ELF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) n

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

!     Read A from data file
      Read (nin,*)(a(i,1:n),i=1,n)

!     Find f( A )
      ifail = 0
      Call f01elf(n,a,lda,fcos2,iuser,ruser,iflag,imnorm,ifail)

!     Print solution
      ifail = 0
      Call x04caf('G','N',n,n,a,lda,'F(A) = COS(2A)',ifail)

!     Print the norm of the imaginary part to check it is small
      Write (nout,*)
      Write (nout,Fmt='(1X,A,F6.2)') 'Imnorm =', imnorm

    End Program f01elfe
```

9.2 Program Data

```
F01ELF Example Program Data

4                            :Value of N

 3.0     0.0     1.0     2.0
-1.0     1.0     3.0     1.0
 0.0     2.0     2.0     1.0
 2.0     1.0    -1.0     1.0  :End of matrix A
```

9.3 Program Results

```
 F01ELF Example Program Results

 F(A) = COS(2A)
             1          2          3          4
 1     -0.1704    -1.1597    -0.1878    -0.7307
```

```
 2      -0.3950    -0.4410     0.7606     0.0655
 3      -0.0950    -0.0717     0.0619    -0.4351
 4      -0.1034     0.6424    -1.3964     0.1042

 Imnorm =  0.00
```