NAG Library Routine Document

F01KKF

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

F01KKF computes the Fréchet derivative $L(A,E)$ of the matrix logarithm of the complex n by n matrix A applied to the complex n by n matrix E. The principal matrix logarithm $\log(A)$ is also returned.

2 Specification

```
SUBROUTINE F01KKF (N, A, LDA, E, LDE, IFAIL)
INTEGER                N, LDA, LDE, IFAIL
COMPLEX (KIND=nag_wp)  A(LDA,*), E(LDE,*)
```

3 Description

For a matrix with no eigenvalues on the closed negative real line, the principal matrix logarithm $\log(A)$ is the unique logarithm whose spectrum lies in the strip $\{z : -\pi < \operatorname{Im}(z) < \pi\}$.

The Fréchet derivative of the matrix logarithm of A is the unique linear mapping $E \mapsto L(A,E)$ such that for any matrix E

$$\log(A+E) - \log(A) - L(A,E) = o(\|E\|).$$

The derivative describes the first order effect of perturbations in A on the logarithm $\log(A)$.

F01KKF uses the algorithm of Al–Mohy *et al.* (2012) to compute $\log(A)$ and $L(A,E)$. The principal matrix logarithm $\log(A)$ is computed using a Schur decomposition, a Padé approximant and the inverse scaling and squaring method. The Padé approximant is then differentiated in order to obtain the Fréchet derivative $L(A,E)$. If A is nonsingular but has negative real eigenvalues, the principal logarithm is not defined, but F01KKF will return a non-principal logarithm and Fréchet derivative.

4 References

Al–Mohy A H and Higham N J (2011) Improved inverse scaling and squaring algorithms for the matrix logarithm *SIAM J. Sci. Comput.* **34(4)** C152–C169

Al–Mohy A H, Higham N J and Relton S D (2012) Computing the Fréchet derivative of the matrix logarithm and estimating the condition number *MIMS EPrint* **2012.72**

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

5 Parameters

1: N – INTEGER *Input*

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

Constraint: $N \geq 0$.

2: 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 matrix A.

On exit: the n by n principal matrix logarithm, $\log(A)$. Alterntively, if IFAIL $= 2$, a non-principal logarithm is returned.

3: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F01KKF is called.

Constraint: LDA $\geq$ N.

4: E(LDE, $*$) – COMPLEX (KIND=nag_wp) array *Input/Output*

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

On entry: the n by n matrix E

On exit: with IFAIL $= 0$, 2 or 3, the Fréchet derivative $L(A, E)$

5: LDE – INTEGER *Input*

On entry: the first dimension of the array E as declared in the (sub)program from which F01KKF is called.

Constraint: LDE $\geq$ N.

6: 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 is singular so the logarithm cannot be computed.

IFAIL $= 2$

A has eigenvalues on the negative real line. The principal logarithm is not defined in this case, so a non-principal logarithm was returned.

IFAIL $= 3$

$\log(A)$ has been computed using an IEEE double precision Padé approximant, although the arithmetic precision is higher than IEEE double precision.

IFAIL $= 4$

An unexpected internal error occurred. This failure should not occur and suggests that the routine has been called incorrectly.

IFAIL $= -1$

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

IFAIL $= -3$

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

IFAIL $= -5$

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

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

For a normal matrix A (for which $A^{\mathrm{H}}A = AA^{\mathrm{H}}$), the Schur decomposition is diagonal and the computation of the matrix logarithm reduces to evaluating the logarithm of the eigenvalues of A and then constructing $\log(A)$ using the Schur vectors. This should give a very accurate result. In general, however, no error bounds are available for the algorithm. The sensitivity of the computation of $\log(A)$ and $L(A, E)$ is worst when A has an eigenvalue of very small modulus or has a complex conjugate pair of eigenvalues lying close to the negative real axis. See Al–Mohy and Higham (2011), Al–Mohy *et al.* (2012) and Section 11.2 of Higham (2008) for details and further discussion.

8 Parallelism and Performance

F01KKF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F01KKF 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 cost of the algorithm is $O(n^3)$ floating-point operations. The complex allocatable memory required is approximately $5n^2$; see Al–Mohy *et al.* (2012) for further details.

If the matrix logarithm alone is required, without the Fréchet derivative, then F01FJF should be used. If the condition number of the matrix logarithm is required then F01KJF should be used. The real analogue of this routine is F01JKF.

10 Example

This example finds the principal matrix logarithm $\log(A)$ and the Fréchet derivative $L(A,E)$, where

$$A = \begin{pmatrix} 1+4i & 3i & i & 2 \\ 2i & 3 & 1 & 1+i \\ i & 2+i & 2 & i \\ 1+2i & 3+2i & 1+2i & 3+i \end{pmatrix} \quad \text{and} \quad E = \begin{pmatrix} 1 & 1+2i & 2 & 2+i \\ 1+3i & i & 1 & 0 \\ 2i & 4+i & 1 & 1 \\ 1 & 2+2i & 3i & 1 \end{pmatrix}.$$

10.1 Program Text

```
    Program f01kkfe

!     F01KKF Example Program Text
!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: f01kkf, nag_wp, x04daf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ierr, ifail, lda, lde, n
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:,:), e(:,:)
!     .. Executable Statements ..
      Write (nout,*) 'F01KKF Example Program Results'
      Write (nout,*)
      Flush (nout)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) n

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

!     Read A from data file

      Read (nin,*)(a(i,1:n),i=1,n)

!     Read E from data file

      Read (nin,*)(e(i,1:n),i=1,n)

!     ifail: behaviour on error exit
!             =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0

!     Find log( A ) and L_log(A,E)
      Call f01kkf(n,a,lda,e,lde,ifail)

!     Print solution

      ierr = 0
      Call x04daf('General',' ',n,n,a,lda,'Log(A)',ierr)
      Write (nout,*)
      Call x04daf('General',' ',n,n,e,lde,'L_log(A,E)',ierr)

    End Program f01kkfe
```

10.2 Program Data

```
F01KKF Example Program Data

  4                                                 :Value of N

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

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

10.3 Program Results

```
 F01KKF Example Program Results

 Log(A)
              1          2          3          4
 1       1.4188     0.2758    -0.2240     0.4528
         1.2438     1.0040     0.0826    -0.5887

 2       0.2299     1.0702     0.5292     0.1976
         0.4825    -0.3306    -0.0422     0.1532

 3       0.1328     0.9235     0.6051    -0.1211
        -0.0462     0.3060    -0.0973     0.2966

 4       0.4704     1.0779     0.2724     0.9612
        -0.0891     0.0538     0.7627     0.2680

 L_log(A,E)
              1          2          3          4
 1       0.1620    -0.0593    -0.1543     0.5534
        -0.6532     0.8434    -1.3537     0.0869

 2       0.6673     0.0637     0.3421    -0.4639
         0.7351    -0.0911     0.1136    -0.3399

 3      -0.2500     1.4898    -0.1547     0.3319
        -0.0433     0.6186    -0.0495    -0.3078

 4      -0.4004     0.5834    -0.5153     0.4407
        -0.5893    -0.5926     1.4107     0.1236
```