NAG Library Routine Document

F01FJF

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

F01FJF computes the principal matrix logarithm, $\log(A)$, of a complex n by n matrix A, with no eigenvalues on the closed negative real line.

2 Specification

```
SUBROUTINE F01FJF (N, A, LDA, IFAIL)
INTEGER                 N, LDA, IFAIL
COMPLEX (KIND=nag_wp) A(LDA,*)
```

3 Description

Any nonsingular matrix A has infinitely many logarithms. For a matrix with no eigenvalues on the closed negative real line, the principal logarithm is the unique logarithm whose spectrum lies in the strip $\{z : -\pi < \mathrm{Im}(z) < \pi\}$. If A is nonsingular but has eigenvalues on the negative real line, the principal logarithm is not defined, but F01FJF will return a non-principal logarithm.

$\log(A)$ is computed using the inverse scaling and squaring algorithm for the matrix logarithm described in Al–Mohy and Higham (2011).

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

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

5 Arguments

1: N – INTEGER *Input*

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

Constraint: N ≥ 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, $\log(A)$, unless IFAIL $= 2$, in which case 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 F01FJF is called.

Constraint: LDA $\geq$ N.

4: 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 $= 1$

A is singular so the logarithm cannot be computed.

IFAIL $= 2$

A was found to have 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 has occurred. Please contact NAG.

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 $= -99$

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

For a normal matrix A (for which $A^{\mathrm{H}}A = AA^{\mathrm{H}}$), the Schur decomposition is diagonal and the algorithm 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. See Al–Mohy and Higham (2011) and Section 9.4 of Higham (2008) for details and further discussion.

The sensitivity of the computation of $\log(A)$ 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.

If estimates of the condition number of the matrix logarithm are required then F01KJF should be used.

8 Parallelism and Performance

F01FJF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F01FJF 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 $O\left(n^3\right)$ floating-point operations (see Al–Mohy and Higham (2011)). The complex allocatable memory required is approximately $3 \times n^2$.

If the Fréchet derivative of the matrix logarithm is required then F01KKF should be used.

F01EJF can be used to find the principal logarithm of a real matrix.

10 Example

This example finds the principal matrix logarithm of the matrix

$$A = \begin{pmatrix} 1.0+2.0i & 0.0+1.0i & 1.0+0.0i & 3.0+2.0i \\ 0.0+3.0i & -2.0+0.0i & 0.0+0.0i & 1.0+0.0i \\ 1.0+0.0i & -2.0+0.0i & 3.0+2.0i & 0.0+3.0i \\ 2.0+0.0i & 0.0+1.0i & 0.0+1.0i & 2.0+3.0i \end{pmatrix}.$$

10.1 Program Text

```
    Program f01fjfe

!     F01FJF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: f01fjf, nag_wp, x04daf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ifail, lda, n
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:,:)
!     .. Executable Statements ..
      Write (nout,*) 'F01FJF Example Program Results'
      Write (nout,*)
      Flush (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)

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

!     Find log( A )
      Call f01fjf(n,a,lda,ifail)

!     Print solution
      ifail = 0
      Call x04daf('G','N',n,n,a,lda,'log(A)',ifail)

    End Program f01fjfe
```

10.2 Program Data

```
F01FJF Example Program Data

   4                                                 :Value of N

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

10.3 Program Results

```
 F01FJF Example Program Results

 log(A)
                1          2          3          4
 1       1.0390     0.2859     0.0516     0.7586
         1.1672     0.3998    -0.2562    -0.4678

 2      -2.7481     1.1898     0.1369     2.1771
         2.6187    -2.2287    -0.9128    -1.0118

 3      -0.8514    -0.2517     1.3839     1.1920
         0.3927    -0.4791     0.2129     0.4240

 4       1.1970    -0.6813     0.0051     0.7867
        -0.1242     0.3969     0.3511     0.7502
```