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 Schur–Parlett algorithm for the matrix logarithm described in Higham (2008) and Davies and Higham (2003).

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

5 Parameters

1: N – INTEGER *Input*

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

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

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, $\log(A)$, unless IFAIL $= 4$, 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: $\mathrm{LDA} \geq \max(1, \mathrm{N})$.

4: 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 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 $= 3$

The arithmetic precision is higher than that used for the Padé approximant computed matrix logarithm.

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 $= -999$

Allocation of memory failed. The complex allocatable memory required is approximately 3N^2.

7 Accuracy

For a normal matrix A (for which $A^{\text{H}}A = AA^{\text{H}}$), 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 Section 9.4 of Higham (2008) for details and further discussion.

For discussion of the condition of the matrix logarithm see Section 11.2 of Higham (2008). In particular, the condition number of the matrix logarithm at A, $\kappa_{\log}(A)$, which is a measure of the sensitivity of the computed logarithm to perturbations in the matrix A, satisfies

$$\kappa_{\log}(A) \geq \frac{\kappa(A)}{\|\log(A)\|},$$

where $\kappa(A)$ is the condition number of A. Further, 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.

8 Further Comments

Up to $4n^2$ of complex allocatable memory may be required.

The cost of the algorithm is $O\left(n^3\right)$ floating point operations. The exact cost depends on the eigenvalue distribution of A; see Algorithm 11.11 of Higham (2008).

If estimates of the condition number of the matrix logarithm are required then F01KAF should be used.

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

9 Example

This example finds the principal matrix logarithm of the matrix

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

9.1 Program Text

```
    Program f01fjfe

!     F01FJF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. 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,*)
!     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 log( A )
      ifail = 0
      Call f01fjf(n,a,lda,ifail)
```

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

    End Program f01fjfe
```

9.2 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
```

9.3 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
```