NAG Library Routine Document

F01FCF

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

F01FCF computes the matrix exponential, e^A, of a complex n by n matrix A.

2 Specification

```
SUBROUTINE F01FCF (N, A, LDA, IFAIL)

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

3 Description

e^A is computed using a Padé approximant and the scaling and squaring method described in Higham (2005) and Higham (2008).

If A has a full set of eigenvectors V then A can be factorized as

$$A = VDV^{-1},$$

where D is the diagonal matrix whose diagonal elements, d_i, are the eigenvalues of A. e^A is then given by

$$e^A = Ve^DV^{-1},$$

where e^D is the diagonal matrix whose ith diagonal element is e^{d_i}.

Note that e^A is not computed this way as to do so would, in general, be unstable.

4 References

Higham N J (2005) The scaling and squaring method for the matrix exponential revisited *SIAM J. Matrix Anal. Appl.* **26(4)** 1179–1193

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

Moler C B and Van Loan C F (2003) Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later *SIAM Rev.* **45** 3–49

5 Parameters

1: N – INTEGER *Input*

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

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

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: with $\text{IFAIL} = 0$, the n by n matrix exponential, e^A.

3: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F01FCF is called.

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

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: $\text{IFAIL} = 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry $\text{IFAIL} = 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

$\text{IFAIL} < 0$ and $\text{IFAIL} \neq -999$

If $\text{IFAIL} = -i$, the ith argument had an illegal value.

$\text{IFAIL} = -999$

Internal memory allocation failed.

The INTEGER allocatable memory required is N, and the complex allocatable memory required is approximately $6 \times \text{N}^2$.

$\text{IFAIL} > 0$ and $\text{IFAIL} \leq \text{N} + 1$

Note:these failures should not occur, and suggest that the routine has been called incorrectly.

If $\text{IFAIL} \leq \text{N}$, the linear equations to be solved for the Padé approximant are singular.

If $\text{IFAIL} = \text{N} + 1$, the linear equations to be solved are nearly singular and the Padé approximant probably has no correct figures.

$\text{IFAIL} = \text{N} + 2$

e^A has been computed using an IEEE double precision Padé approximant, although the arithmetic precision is higher than IEEE double precision.

7 Accuracy

For a normal matrix A (for which $A^{\text{H}}A = AA^{\text{H}}$) the computed matrix, e^A, is guaranteed to be close to the exact matrix, that is, the method is forward stable. No such guarantee can be given for non-normal matrices. See Section 10.3 of Higham (2008) for details and further discussion.

For discussion of the condition of the matrix exponential see Section 10.2 of Higham (2008).

8 Further Comments

The cost of the algorithm is $O(n^3)$; see Algorithm 10.20 in Higham (2008).

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

As well as the excellent book cited above, the classic reference for the computation of the matrix exponential is Moler and Van Loan (2003).

9 Example

This example find the matrix exponential of the matrix

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

9.1 Program Text

```
    Program f01fcfe

!     F01FCF Example Program Text
!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: f01fcf, nag_wp, x04daf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter              :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                         :: i, ierr, ifail, lda, n
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:,:)
!     .. Executable Statements ..
      Write (nout,*) 'F01FCF 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)

!     Find exp( A )

      ifail = 0
      Call f01fcf(n,a,lda,ifail)

!     Print solution

      ierr = 0
      Call x04daf('General',' ',n,n,a,lda,'Exp(A)',ierr)

    End Program f01fcfe
```

9.2 Program Data

```
F01FCF Example Program Data

  4                                             :Value of N

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

9.3 Program Results

```
 F01FCF Example Program Results

 Exp(A)
              1          2          3          4
 1   -157.9003  -194.6526  -186.5627  -155.7669
     -754.3717  -555.0507  -475.4533  -520.1876

 2   -206.8899  -225.4985  -212.4414  -186.5627
     -694.7443  -505.3938  -431.0611  -475.4533

 3   -208.7476  -238.4962  -225.4985  -194.6526
     -808.2090  -590.8045  -505.3938  -555.0507

 4   -133.3958  -208.7476  -206.8899  -157.9003
    -1085.5496  -808.2090  -694.7443  -754.3717
```