NAG Library Routine Document

F01ECF

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

F01ECF computes the matrix exponential, e^A, of a real n by n matrix A.

2 Specification

```
SUBROUTINE F01ECF(N, A, LDA, IFAIL)
INTEGER              N, LDA, IFAIL
double precision     A(LDA,N)
```

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 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 (1978) Nineteen dubious ways to compute the exponential of a matrix *SIAM Review* **20** 801–836

5 Parameters

1: N – INTEGER *Input*

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

Constraint: $N \geq 0$.

2: A(LDA,N) – ***double precision*** array *Input/Output*

On entry: the n by n matrix A.

On exit: 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 F01ECF 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.

On exit: $\mathrm{IFAIL} = 0$ unless the routine detects an error (see Section 6).

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.**

6 Error Indicators and Warnings

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

Errors or warnings detected by the routine:

$\mathrm{IFAIL} < 0$ and $\mathrm{IFAIL} \geq -4$

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

$\mathrm{IFAIL} = -999$

Allocation of memory failed. The INTEGER allocatable memory required is N, and the ***double precision*** allocatable memory required is approximately $6 \times \mathrm{N}^2$.

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

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

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

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

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

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 $A^{\mathrm{H}}A = AA^{\mathrm{H}}$) 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 of Higham (2008).

As well as the excellent book cited above, the classic reference for the computation of the matrix exponential is Moler and Van Loan (1978).

9 Example

This example finds the matrix exponential of the matrix

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

9.1 Program Text

```
*     F01ECF Example Program Text
*     Mark 22 Release. NAG Copyright 2008.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=10)
      INTEGER          LDA
      PARAMETER        (LDA=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IERR, IFAIL, J, N
*     .. Local Arrays ..
      DOUBLE PRECISION A(LDA,NMAX)
*     .. External Subroutines ..
      EXTERNAL         F01ECF, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F01ECF Example Program Results'
      WRITE (NOUT,*)
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) ((A(I,J),J=1,N),I=1,N)
*
*        Find exp( A )
*
         IFAIL = -1
         CALL F01ECF(N,A,LDA,IFAIL)
*
         IF (IFAIL.EQ.0) THEN
*
*           Print solution
*
            IERR = 0
            CALL X04CAF('General',' ',N,N,A,LDA,'Exp(A)',IERR)
*
         ELSE
*
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'Failure in F01ECF'
            WRITE (NOUT,99999) IFAIL
         END IF
      ELSE
         WRITE (NOUT,*) 'NMAX too small'
      END IF
*
99999 FORMAT (6X,'IFAIL = ',I5)
      END
```

9.2 Program Data

```
F01ECF Example Program Data

  4                      :Value of N

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

9.3 Program Results

```
 F01ECF Example Program Results

 Exp(A)
             1           2           3           4
 1    740.7038    610.8500    542.2743    549.1753
 2    731.2510    603.5524    535.0884    542.2743
 3    823.7630    679.4257    603.5524    610.8500
 4    998.4355    823.7630    731.2510    740.7038
```