NAG Library Routine Document

F01EDF

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

F01EDF computes the matrix exponential, e^A, of a real symmetric n by n matrix A.

2 Specification

```
SUBROUTINE F01EDF (UPLO, N, A, LDA, IFAIL)

INTEGER              N, LDA, IFAIL
REAL (KIND=nag_wp) A(LDA,*)
CHARACTER(1)         UPLO
```

3 Description

e^A is computed using a spectral factorization of A

$$A = QDQ^{\mathrm{T}},$$

where D is the diagonal matrix whose diagonal elements, d_i, are the eigenvalues of A, and Q is an orthogonal matrix whose columns are the eigenvectors of A. e^A is then given by

$$e^A = Qe^DQ^{\mathrm{T}},$$

where e^D is the diagonal matrix whose ith diagonal element is e^{d_i}. See for example Section 4.5 of Higham (2008).

4 References

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: UPLO – CHARACTER(1) *Input*

On entry: indicates whether the upper or lower triangular part of A is stored.

UPLO = 'U'
The upper triangular part of A is stored.

UPLO = 'L'
The lower triangular part of A is stored.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER *Input*

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

Constraint: N ≥ 0.

3: $\mathrm{A}(\mathrm{LDA},*)$ – REAL (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 symmetric matrix A.

If UPLO = 'U', the upper triangular part of A must be stored and the elements of the array below the diagonal are not referenced.

If UPLO = 'L', the lower triangular part of A must be stored and the elements of the array above the diagonal are not referenced.

On exit: if IFAIL $= 0$, the upper or lower triangular part of the n by n matrix exponential, e^A.

4: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F01EDF is called.

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

5: 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 > 0

The computation of the spectral factorization failed to converge.

IFAIL $= -1$

On entry, UPLO $= \langle$*value*$\rangle$.
Constraint: UPLO = 'L' or 'U'.

IFAIL $= -2$

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

IFAIL $= -3$

An internal error occurred when computing the spectral factorization. Please contact NAG.

IFAIL $= -4$

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 3.8 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 symmetric matrix A, the matrix e^A, has the relative condition number

$$\kappa(A) = \|A\|_2,$$

which is the minimum possible for the matrix exponential and so the computed matrix exponential is guaranteed to be close to the exact matrix. See Section 10.2 of Higham (2008) for details and further discussion.

8 Parallelism and Performance

F01EDF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F01EDF 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 integer allocatable memory required is N, and the real allocatable memory required is approximately $(\mathrm{N} + nb + 4) \times \mathrm{N}$, where nb is the block size required by F08FAF (DSYEV).

The cost of the algorithm is $O(n^3)$.

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

10 Example

This example finds the matrix exponential of the symmetric matrix

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 2 & 3 \\ 3 & 2 & 1 & 2 \\ 4 & 3 & 2 & 1 \end{pmatrix}$$

10.1 Program Text

```
    Program f01edfe

!     F01EDF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: f01edf, nag_wp, x04caf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ifail, lda, n
      Character (1)                    :: uplo
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:,:)
!     .. Executable Statements ..
      Write (nout,*) 'F01EDF Example Program Results'
      Write (nout,*)
      Flush (nout)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) n
      Read (nin,*) uplo
      lda = n
      Allocate (a(lda,n))
!     Read A from data file
      If (uplo=='U' .Or. uplo=='u') Then
        Read (nin,*)(a(i,i:n),i=1,n)
      Else
        Read (nin,*)(a(i,1:i),i=1,n)
      End If

!     ifail: behaviour on error exit
!            =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
!     Find exp( A )
      Call f01edf(uplo,n,a,lda,ifail)

!     Print solution
      Call x04caf(uplo,'N',n,n,a,lda,'Symmetric Exp(A)',ifail)

    End Program f01edfe
```

10.2 Program Data

```
F01EDF Example Program Data

  4                     : n
 'U'                    : uplo

  1.0   2.0   3.0   4.0
        1.0   2.0   3.0
              1.0   2.0
                    1.0  : a
```

10.3 Program Results

```
F01EDF Example Program Results

Symmetric Exp(A)
           1          2          3          4
1  2675.3899  2193.0210  2193.2062  2675.2803
2             1798.3297  1797.8497  2193.2062
3                        1798.3297  2193.0210
4                                   2675.3899
```