NAG Library Routine Document

F01EKF

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

F01EKF computes the matrix exponential, sine, cosine, sinh or cosh, of a real n by n matrix A using the Schur–Parlett algorithm.

2 Specification

```
SUBROUTINE F01EKF (FUN, N, A, LDA, IMNORM, IFAIL)
INTEGER             N, LDA, IFAIL
REAL (KIND=nag_wp) A(LDA,*), IMNORM
CHARACTER(*)        FUN
```

3 Description

$f(A)$, where f is either the exponential, sine, cosine, sinh or cosh, is computed using the Schur–Parlett algorithm 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: FUN – CHARACTER(*) *Input*

On entry: indicates which matrix function will be computed.

FUN = 'EXP'
The matrix exponential, e^A, will be computed.

FUN = 'SIN'
The matrix sine, $\sin(A)$, will be computed.

FUN = 'COS'
The matrix cosine, $\cos(A)$, will be computed.

FUN = 'SINH'
The hyperbolic matrix sine, $\sinh(A)$, will be computed.

FUN = 'COSH'
The hyperbolic matrix cosine, $\cosh(A)$, will be computed.

Constraint: FUN = 'EXP', 'SIN', 'COS', 'SINH' or 'COSH'.

2: N – INTEGER *Input*

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

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

3: A(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 matrix A.

On exit: the n by n matrix, $f(A)$.

4: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F01EKF is called.

Constraint: LDA $\geq$ N.

5: IMNORM – REAL (KIND=nag_wp) *Output*

On exit: if A has complex eigenvalues, F01EKF will use complex arithmetic to compute the matrix function. The imaginary part is discarded at the end of the computation, because it will theoretically vanish. IMNORM contains the 1-norm of the imaginary part, which should be used to check that the routine has given a reliable answer.

If A has real eigenvalues, F01EKF uses real arithmetic and IMNORM $= 0$.

6: 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 Taylor series failed to converge.

IFAIL $= 2$

An unexpected internal error occurred when evaluating the function at a point. Please contact NAG.

IFAIL $= 3$

There was an error whilst reordering the Schur form of A.
Note: this failure should not occur and suggests that the routine has been called incorrectly.

IFAIL $= 4$

The routine was unable to compute the Schur decomposition of A.
Note: this failure should not occur and suggests that the routine has been called incorrectly.

IFAIL $= 5$

An unexpected internal error occurred. Please contact NAG.

IFAIL $= 6$

The linear equations to be solved are nearly singular and the Padé approximant used to compute the exponential may have no correct figures.
Note: this failure should not occur and suggests that the routine has been called incorrectly.

IFAIL $= -1$

On entry, FUN $= \langle value \rangle$ was an illegal value.

IFAIL $= -2$

Input parameter number $\langle value \rangle$ is invalid.

IFAIL $= -4$

On entry, parameter LDA is invalid.
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 normal matrix A (for which $A^{\mathrm{T}}A = AA^{\mathrm{T}}$), the Schur decomposition is diagonal and the algorithm reduces to evaluating f at the eigenvalues of A and then constructing $f(A)$ using the Schur vectors. This should give a very accurate result. In general, however, no error bounds are available for the algorithm.

For further discussion of the Schur–Parlett algorithm see Section 9.4 of Higham (2008).

8 Parallelism and Performance

F01EKF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library. In these implementations, this routine may make calls to the user-supplied functions from within an OpenMP parallel region. Thus OpenMP directives within the user functions can only be used if you are compiling the user-supplied function and linking the executable in accordance with the instructions in the Users' Note for your implementation.

F01EKF 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. If A has real eigenvalues then up to $9n^2$ of real allocatable memory may be required. If A has complex eigenvalues then up to $9n^2$ of complex allocatable memory may be required.

The cost of the Schur–Parlett algorithm depends on the spectrum of A, but is roughly between $28n^3$ and $n^4/3$ floating-point operations; see Algorithm 9.6 of Higham (2008).

If the matrix exponential is required then it is recommended that F01ECF be used. F01ECF uses an algorithm which is, in general, more accurate than the Schur–Parlett algorithm used by F01EKF.

If estimates of the condition number of the matrix function are required then F01JAF should be used.

F01FKF can be used to find the matrix exponential, sin, cos, sinh or cosh of a complex matrix.

10 Example

This example finds the matrix cosine of the matrix

$$A = \begin{pmatrix} 2 & 0 & 1 & 0 \\ 0 & 2 & -2 & 1 \\ 0 & 2 & 3 & 1 \\ 1 & 4 & 0 & 0 \end{pmatrix}.$$

10.1 Program Text

```
    Program f01ekfe

!     F01EKF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: f01ekf, nag_wp, x04caf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: imnorm
      Integer                          :: i, ifail, lda, n
      Character (4)                    :: fun
      Character (20)                   :: fun_name
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:,:)
!     .. Executable Statements ..
      Write (nout,*) 'F01EKF Example Program Results'
      Write (nout,*)
      Flush (nout)

!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) n, fun

      lda = n
      Allocate (a(lda,n))

!     Read A from data file
      Read (nin,*)(a(i,1:n),i=1,n)

!     Find f( A )
      ifail = 0
      Call f01ekf(fun,n,a,lda,imnorm,ifail)

!     Print solution
      Write (fun_name,Fmt='(3(A))') 'F(A) = ', fun, '(A)'
      Write (nout,*)
```

```
      ifail = 0
      Call x04caf('G','N',n,n,a,lda,fun_name,ifail)

!     Print the norm of the imaginary part to check it is small
      Write (nout,*)
      Write (nout,Fmt='(1X,A,F6.2)') 'Imnorm =', imnorm

    End Program f01ekfe
```

10.2 Program Data

```
F01EKF Example Program Data

4       COS                  :Values of N and FUN

2.0     0.0     1.0     0.0
0.0     2.0    -2.0     1.0
0.0     2.0     3.0     1.0
1.0     4.0     0.0     0.0  :End of matrix A
```

10.3 Program Results

```
 F01EKF Example Program Results


 F(A) = COS (A)
              1          2          3          4
 1     -0.2998     1.5003    -0.7849     0.4677
 2     -0.2385    -3.2657     0.5812    -1.1460
 3      0.4677     0.3008    -4.0853    -0.2200
 4     -0.2107    -2.8199    -1.2964    -0.8325

 Imnorm =  0.00
```