NAG Library Routine Document

F01FKF

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

F01FKF computes the matrix exponential, sine, cosine, sinh or cosh, of a complex n by n matrix A using the Schur–Parlett algorithm.

2 Specification

```
SUBROUTINE F01FKF (FUN, N, A, LDA, IFAIL)

INTEGER              N, LDA, IFAIL
COMPLEX (KIND=nag_wp) A(LDA,*)
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: $N \geq 0$.

3: 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 matrix, $f(A)$.

4: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F01FKF is called.

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

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 $= 1$

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 $= -1$

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

IFAIL $= -2$

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

IFAIL $= -4$

On entry, parameter LDA is invalid.
Constraint: LDA $\geq$ N.

IFAIL $= -999$

Allocation of memory failed.

7   Accuracy

For a normal matrix A (for which $A^{\mathrm{H}}A = AA^{\mathrm{H}}$), 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   Further Comments

The integer allocatable memory required is n, and the complex allocatable memory required is approximately $9n^2$.

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 F01FCF be used. F01FCF uses an algorithm which is, in general, more accurate than the Schur–Parlett algorithm used by F01FKF.

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

F01EKF can be used to find the matrix exponential, sin, cos, sinh or cosh of a real matrix A.

9   Example

This example finds the matrix sinh of the matrix

$$A = \begin{pmatrix} 1.0 + 1.0i & 0.0 + 0.0i & 1.0 + 3.0i & 0.0 + 0.0i \\ 0.0 + 0.0i & 2.0 + 0.0i & 0.0 + 0.0i & 1.0 + 2.0i \\ 3.0 + 1.0i & 0.0 + 4.0i & 1.0 + 1.0i & 0.0 + 0.0i \\ 1.0 + 1.0i & 0.0 + 2.0i & 0.0 + 0.0i & 1.0 + 0.0i \end{pmatrix}.$$

9.1   Program Text

```
Program f01fkfe

!     F01FKF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: f01fkf, nag_wp, x04daf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter           :: nin = 5, nout = 6
!     .. Local Scalars ..
```

```
      Integer                         :: i, ifail, lda, n
      Character (4)                   :: fun
      Character (20)                  :: fun_name
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:,:)
!     .. Executable Statements ..
      Write (nout,*) 'F01FKF Example Program Results'
      Write (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 f01fkf(fun,n,a,lda,ifail)

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

      ifail = 0
      Call x04daf('G','N',n,n,a,lda,fun_name,ifail)

   End Program f01fkfe
```

9.2   Program Data

```
F01FKF Example Program Data

   4          SINH                                    :Values of N and FUN

  (1.0, 1.0) ( 0.0, 0.0) ( 1.0, 3.0) (0.0, 0.0)
  (0.0, 0.0) ( 2.0, 0.0) ( 0.0, 0.0) (1.0, 2.0)
  (3.0, 1.0) ( 0.0, 4.0) ( 1.0, 1.0) (0.0, 0.0)
  (1.0, 1.0) ( 0.0, 2.0) ( 0.0, 0.0) (1.0, 0.0)  :End of matrix A
```

9.3   Program Results

```
 F01FKF Example Program Results


 F(A) = SINH(A)
            1          2          3          4
 1     -4.3015    -1.4918    -4.4242     1.4438
       -1.8117    -8.7793    -1.3925    -6.5287

 2     -1.7976     1.4211    -1.2712     1.2118
       -0.2935    -0.1993    -1.9931     2.8506

 3     -4.4968    -5.7934    -4.3015    -3.0082
       -0.1964    -4.7166    -1.8117    -4.1821

 4     -2.1506    -0.6103    -1.5163     0.0385
       -0.3911    -1.4408    -1.9317    -0.2847
```