NAG Library Routine Document

F01HAF

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

F01HAF computes the action of the matrix exponential e^{tA}, on the matrix B, where A is a complex n by n matrix, B is a complex n by m matrix and t is a complex scalar.

2 Specification

```
SUBROUTINE F01HAF (N, M, A, LDA, B, LDB, T, IFAIL)
INTEGER                 N, M, LDA, LDB, IFAIL
COMPLEX (KIND=nag_wp) A(LDA,*), B(LDB,*), T
```

3 Description

$e^{tA}B$ is computed using the algorithm described in Al–Mohy and Higham (2011) which uses a truncated Taylor series to compute the product $e^{tA}B$ without explicitly forming e^{tA}.

4 References

Al–Mohy A H and Higham N J (2011) Computing the action of the matrix exponential, with an application to exponential integrators *SIAM J. Sci. Statist. Comput.* **33(2)** 488-511

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

5 Parameters

1: N – INTEGER *Input*

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

Constraint: N ≥ 0.

2: M – INTEGER *Input*

On entry: m, the number of columns of the matrix B.

Constraint: M ≥ 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: A is overwritten during the computation.

4: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F01HAF is called.

Constraint: LDA $\geq$ N.

5: B(LDB, $*$) – COMPLEX (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array B must be at least M.

On entry: the n by m matrix B.

On exit: the n by m matrix $e^{tA}B$.

6: LDB – INTEGER *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F01HAF is called.

Constraint: LDB $\geq$ N.

7: T – COMPLEX (KIND=nag_wp) *Input*

On entry: the scalar t.

8: 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 $= 2$

$e^{tA}B$ has been computed using an IEEE double precision Taylor series, although the arithmetic precision is higher than IEEE double precision.

IFAIL $= -1$

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

IFAIL $= -2$

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

IFAIL $= -4$

On entry, LDA $= \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: LDA $\geq$ N.

IFAIL $= -6$

On entry, LDB $= \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: LDB $\geq$ 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 Hermitian matrix A (for which $A^{\mathrm{H}} = A$) the computed matrix $e^{tA}B$ is guaranteed to be close to the exact matrix, that is, the method is forward stable. No such guarantee can be given for non-Hermitian matrices. See Section 4 of Al–Mohy and Higham (2011) for details and further discussion.

8 Parallelism and Performance

F01HAF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F01HAF 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 matrix $e^{tA}B$ could be computed by explicitly forming e^{tA} using F01FCF and multiplying B by the result. However, experiments show that it is usually both more accurate and quicker to use F01HAF.

The cost of the algorithm is $O(n^2m)$. The precise cost depends on A since a combination of balancing, shifting and scaling is used prior to the Taylor series evaluation.

Approximately $n^2 + (2m + 8)n$ of complex allocatable memory is required by F01HAF.

F01GAF can be used to compute $e^{tA}B$ for real A, B, and t. F01HBF provides an implementation of the algorithm with a reverse communication interface, which returns control to the user when matrix multiplications are required. This should be used if A is large and sparse.

10 Example

This example computes $e^{tA}B$, where

$$A = \begin{pmatrix} 0.5 + 0.0i & -0.2 + 0.0i & 1.0 + 0.1i & 0.0 + 0.4i \\ 0.3 + 0.0i & 0.5 + 1.2i & 3.1 + 0.0i & 1.0 + 0.2i \\ 0.0 + 2.0i & 0.1 + 0.0i & 1.2 + 0.2i & 0.5 + 0.0i \\ 1.0 + 0.3i & 0.0 + 0.2i & 0.0 + 0.9i & 0.5 + 0.0i \end{pmatrix},$$

$$B = \begin{pmatrix} 0.4 + 0.0i & 1.2 + 0.0i \\ 1.3 + 0.0i & -0.2 + 0.1i \\ 0.0 + 0.3i & 2.1 + 0.0i \\ 0.4 + 0.0i & -0.9 + 0.0i \end{pmatrix}$$

and

$$t = -0.5 + 0.0i.$$

10.1 Program Text

```
    Program f01hafe

!     F01HAF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: f01haf, nag_wp, x04daf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Complex (Kind=nag_wp)            :: t
      Integer                          :: i, ifail, lda, ldb, m, n
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:,:), b(:,:)
!     .. Executable Statements ..
      Write (nout,*) 'F01HAF Example Program Results'
      Write (nout,*)
      Flush (nout)

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

      lda = n
      ldb = n

      Allocate (a(lda,n))
      Allocate (b(ldb,m))

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

!     Read B from data file
      Read (nin,*)(b(i,1:m),i=1,n)

!     Find exp(tA)B
      ifail = 0
      Call f01haf(n,m,a,lda,b,ldb,t,ifail)

      If (ifail==0) Then
!       Print solution
        ifail = 0
        Call x04daf('G','N',n,m,b,ldb,'exp(tA)B',ifail)
      End If
    End Program f01hafe
```

10.2 Program Data

```
F01HAF Example Program Data

4    2    (-0.5,0.0)                              :Values of N, M and T

(0.5,0.0)    (-0.2,0.0)    (1.0,0.1)    (0.0,0.4)
(0.3,0.0)    ( 0.5,1.2)    (3.1,0.0)    (1.0,0.2)
(0.0,2.0)    ( 0.1,0.0)    (1.2,0.2)    (0.5,0.0)
(1.0,0.3)    ( 0.0,0.2)    (0.0,0.9)    (0.5,0.0) :End of matrix A

(0.4,0.0)    ( 1.2,0.0)
(1.3,0.0)    (-0.2,0.1)
(0.0,0.3)    ( 2.1,0.0)
(0.4,0.0)    (-0.9,0.0)                              :End of matrix B
```

10.3 Program Results

```
 F01HAF Example Program Results

 exp(tA)B
                1          2
 1       0.4251    -0.0220
        -0.1061     0.3289

 2       0.7229    -1.7931
        -0.5940     1.4952

 3      -0.1394     1.4781
        -0.1151    -0.4514

 4       0.1054    -1.0059
        -0.0786    -0.7079
```