NAG Toolbox

nag_matop_complex_gen_matrix_actexp (f01ha)

1 Purpose

nag_matop_complex_gen_matrix_actexp (f01ha) 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 Syntax

```
[a, b, ifail] = nag_matop_complex_gen_matrix_actexp(m, a, b, t, 'n', n)
[a, b, ifail] = f01ha(m, a, b, t, 'n', n)
```

Note: the interface to this routine has changed since earlier releases of the toolbox:

At Mark 25: **m** was made optional.

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

5.1 Compulsory Input Parameters

1: **m** – INTEGER

m, the number of columns of the matrix B.

Constraint: $\mathbf{m} \geq 0$.

2: **a**$(lda, :)$ – COMPLEX (KIND=nag_wp) array

The first dimension of the array **a** must be at least **n**.

The second dimension of the array **a** must be at least **n**.

The n by n matrix A.

3: **b**$(ldb, :)$ – COMPLEX (KIND=nag_wp) array

The first dimension of the array **b** must be at least **n**.

The second dimension of the array **b** must be at least **m**.

The n by m matrix B.

4: **t** – COMPLEX (KIND=nag_wp)

The scalar t.

5.2 Optional Input Parameters

1: **n** – INTEGER

Default: the first dimension of the arrays **a**, **b**. (An error is raised if these dimensions are not equal.)

n, the order of the matrix A.

Constraint: $\mathbf{n} \geq 0$.

5.3 Output Parameters

1: **a**$(lda, :)$ – COMPLEX (KIND=nag_wp) array

The first dimension of the array **a** will be **n**.

The second dimension of the array **a** will be **n**.

A is overwritten during the computation.

2: **b**$(ldb, :)$ – COMPLEX (KIND=nag_wp) array

The first dimension of the array **b** will be **n**.

The second dimension of the array **b** will be **m**.

The n by m matrix $e^{tA}B$.

3: **ifail** – INTEGER

ifail $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 2$ (*warning*)

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

ifail $= -1$

Constraint: $\mathbf{n} \geq 0$.

ifail $= -2$

Constraint: $\mathbf{m} \geq 0$.

ifail $= -4$

Constraint: $lda \geq \mathbf{n}$.

ifail $= -6$

Constraint: $ldb \geq \mathbf{n}$.

ifail $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

ifail $= -999$

Dynamic memory allocation failed.

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 Further Comments

The matrix $e^{tA}B$ could be computed by explicitly forming e^{tA} using nag_matop_complex_gen_matrix_exp (f01fc) and multiplying B by the result. However, experiments show that it is usually both more accurate and quicker to use nag_matop_complex_gen_matrix_actexp (f01ha).

The cost of the algorithm is $O\left(n^2m\right)$. 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 nag_matop_complex_gen_matrix_actexp (f01ha).

nag_matop_real_gen_matrix_actexp (f01ga) can be used to compute $e^{tA}B$ for real A, B, and t. nag_matop_complex_gen_matrix_actexp_rcomm (f01hb) 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.

9 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

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

9.1 Program Text

```
function f01ha_example

fprintf('f01ha example results\n\n');

a = [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];
b = [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];
```

```
t = complex(-0.5);

% Compute exp(ta)b

[a, exptab, ifail] = f01ha(a, b, t);

disp('exp(tA)B');
disp(exptab);
```

9.2 Program Results

```
     f01ha example results

exp(tA)B
   0.4251 - 0.1061i  -0.0220 + 0.3289i
   0.7229 - 0.5940i  -1.7931 + 1.4952i
  -0.1394 - 0.1151i   1.4781 - 0.4514i
   0.1054 - 0.0786i  -1.0059 - 0.7079i
```