NAG Toolbox

nag_matop_complex_herm_matrix_exp (f01fd)

1 Purpose

nag_matop_complex_herm_matrix_exp (f01fd) computes the matrix exponential, e^A, of a complex Hermitian n by n matrix A.

2 Syntax

```
[a, ifail] = nag_matop_complex_herm_matrix_exp(uplo, a, 'n', n)
[a, ifail] = f01fd(uplo, a, 'n', n)
```

3 Description

e^A is computed using a spectral factorization of A

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

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

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

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 (2005) The scaling and squaring method for the matrix exponential revisited *SIAM J. Matrix Anal. Appl.* **26(4)** 1179–1193

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

5.1 Compulsory Input Parameters

1: **uplo** – CHARACTER(1)

If **uplo** = 'U', the upper triangle of the matrix A is stored.

If **uplo** = 'L', the lower triangle of the matrix A is stored.

Constraint: **uplo** = 'U' or 'L'.

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 Hermitian 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.

5.2 Optional Input Parameters

1: **n** – INTEGER

Default: the first dimension of the array **a**.

n, the order of the matrix A.

Constraint: **n** ≥ 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**.

If **ifail** $= 0$, the upper or lower triangular part of the n by n matrix exponential, e^A.

2: **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 > 0

The computation of the spectral factorization failed to converge.

ifail $= -1$

On entry, **uplo** was invalid.

ifail $= -2$

Constraint: **n** ≥ 0.

ifail $= -3$

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

ifail $= -4$

Constraint: $lda \geq$ **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 an Hermitian matrix A, the matrix e^A, has the relative condition number

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

which is the minimal 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 Further Comments

The integer allocatable memory required is **n**, the double allocatable memory required is **n** and the complex allocatable memory required is approximately $(\mathbf{n} + nb + 1) \times \mathbf{n}$, where nb is the block size required by nag_lapack_zheev (f08fn).

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

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

9 Example

This example finds the matrix exponential of the Hermitian matrix

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

9.1 Program Text

```
     function f01fd_example

fprintf('f01fd example results\n\n');

uplo = 'u';
a = [1,  2 + 1i,  3 + 2i,  4 + 3i;
     0,  1 + 0i,  2 + 1i,  3 + 2i;
     0,  0,       1 + 0i,  2 + 1i;
     0,  0,       0,       1 + 0i];

% Compute exp(a)
[expa, ifail] = f01fd(uplo, a);

% Display results
[ifail] = x04da(uplo, 'n', expa, 'Hermitian Exp(a)');
```

9.2 Program Results

```
     f01fd example results

 Hermitian Exp(a)
                1            2            3            4
 1    1.1457E+04   8.7983E+03   7.8120E+03   8.3103E+03
      0.0000E+00   2.0776E+03   4.5500E+03   7.8871E+03

 2                 7.1339E+03   6.8242E+03   7.8120E+03
                   0.0000E+00   2.0776E+03   4.5500E+03
```

3	7.1339E+03	8.7983E+03
	0.0000E+00	2.0776E+03
4		1.1457E+04
		0.0000E+00