NAG Toolbox

nag_matop_complex_tri_matrix_sqrt (f01fp)

1 Purpose

nag_matop_complex_tri_matrix_sqrt (f01fp) computes the principal matrix square root, $A^{1/2}$, of a complex upper triangular n by n matrix A.

2 Syntax

```
[a, ifail] = nag_matop_complex_tri_matrix_sqrt(a, 'n', n)
[a, ifail] = f01fp(a, 'n', n)
```

3 Description

A square root of a matrix A is a solution X to the equation $X^2 = A$. A nonsingular matrix has multiple square roots. For a matrix with no eigenvalues on the closed negative real line, the principal square root, denoted by $A^{1/2}$, is the unique square root whose eigenvalues lie in the open right half-plane.

nag_matop_complex_tri_matrix_sqrt (f01fp) computes $A^{1/2}$, where A is an upper triangular matrix. $A^{1/2}$ is also upper triangular.

The algorithm used by nag_matop_complex_tri_matrix_sqrt (f01fp) is described in BjÎrck and Hammarling (1983). In addition a blocking scheme described in Deadman *et al.* (2013) is used.

4 References

BjÎrck Ð and Hammarling S (1983) A Schur method for the square root of a matrix *Linear Algebra Appl.* **52/53** 127–140

Deadman E, Higham N J and Ralha R (2013) Blocked Schur Algorithms for Computing the Matrix Square Root *Applied Parallel and Scientific Computing: 11th International Conference, (PARA 2012, Helsinki, Finland)* P. Manninen and P. Úster, Eds *Lecture Notes in Computer Science* **7782** 171–181 Springer–Verlag

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

5 Parameters

5.1 Compulsory Input Parameters

1: **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 upper triangular matrix A.

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**.

Contains, if **ifail** $= 0$, the n by n principal matrix square root, $A^{1/2}$. Alternatively, if **ifail** $= 1$, contains an n by n non-principal square root of 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 $= 1$

A has negative or semisimple, vanishing eigenvalues. The principal square root is not defined in this case; a non-principal square root is returned.

ifail $= 2$

A has a defective vanishing eigenvalue. The square root cannot be found in this case.

ifail $= 3$

An internal error occurred. It is likely that the function was called incorrectly.

ifail $= -1$

Constraint: **n** ≥ 0.

ifail $= -3$

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

The computed square root $\hat{X}$ satisfies $\hat{X}^2 = A + \Delta A$, where $|\Delta A| \approx O(\epsilon)n|\hat{X}|^2$, where ϵ is ***machine precision***. The order of the change in A is to be interpreted elementwise.

8 Further Comments

The cost of the algorithm is $n^3/3$ complex floating-point operations; see Algorithm 6.3 in Higham (2008). $O(2 \times n^2)$ of complex allocatable memory is required by the function.

If A is a full matrix, then nag_matop_complex_gen_matrix_sqrt (f01fn) should be used to compute the principal square root.

If condition number and residual bound estimates are required, then nag_matop_complex_gen_matrix_cond_sqrt (f01kd) should be used. For further discussion of the condition of the matrix square root see Section 6.1 of Higham (2008).

9 Example

This example finds the principal matrix square root of the matrix

$$A = \begin{pmatrix} 2i & 14+2i & 12+3i & 6+4i \\ 0 & -5+12i & 6+18i & 9+16i \\ 0 & 0 & 3-4i & 16-4i \\ 0 & 0 & 0 & 4 \end{pmatrix}.$$

9.1 Program Text

```
function f01fp_example

fprintf('f01fp example results\n\n');

% Principal square root of complex matrix A

a = [ 2i   14 +  2i   12 +  3i    6 +  4i;
      0    -5 + 12i    6 + 18i    9 + 16i;
      0     0          3 -  4i   16 -  4i;
      0     0          0          4 +  0i];

[as, ifail] = f01fp(a);

disp('Square root of A:');
disp(as);
```

9.2 Program Results

```
f01fp example results

Square root of A:
   1.0000 + 1.0000i   2.0000 - 2.0000i   0.0000 + 1.0000i   1.0000 - 1.0000i
   0.0000 + 0.0000i   2.0000 + 3.0000i   3.0000 + 3.0000i   0.0000 + 1.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   2.0000 - 1.0000i   4.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   2.0000 + 0.0000i
```