NAG Toolbox

nag_matop_real_tri_matrix_sqrt (f01ep)

1 Purpose

nag_matop_real_tri_matrix_sqrt (f01ep) computes the principal matrix square root, $A^{1/2}$, of a real upper quasi-triangular n by n matrix A.

2 Syntax

```
[a, ifail] = nag_matop_real_tri_matrix_sqrt(a, 'n', n)
[a, ifail] = f01ep(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_real_tri_matrix_sqrt (f01ep) computes $A^{1/2}$, where A is an upper quasi-triangular matrix, with 1×1 and 2×2 blocks on the diagonal. Such matrices arise from the Schur factorization of a real general matrix, as computed by nag_lapack_dhseqr (f08pe), for example. nag_matop_real_tri_matrix_sqrt (f01ep) does not require A to be in the canonical Schur form described in nag_lapack_dhseqr (f08pe), it merely requires A to be upper quasi-triangular. $A^{1/2}$ then has the same block triangular structure as A.

The algorithm used by nag_matop_real_tri_matrix_sqrt (f01ep) is described in Higham (1987). In addition a blocking scheme described in Deadman *et al.* (2013) is used.

4 References

BjÎrck Đ 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 (1987) Computing real square roots of a real matrix *Linear Algebra Appl.* **88/89** 405–430

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

5 Parameters

5.1 Compulsory Input Parameters

1: **a**(lda, :) – REAL (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 quasi-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: $\mathbf{n} \geq 0$.

5.3 Output Parameters

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

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

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

The n by n principal matrix square root $A^{1/2}$.

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 vanishing eigenvalues. The principal square root is not defined in this case. nag_matop_real_gen_matrix_sqrt (f01en) or nag_matop_complex_gen_matrix_sqrt (f01fn) may be able to provide further information.

ifail $= 2$

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

ifail $= -1$

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

ifail $= -3$

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

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

The computed square root $\hat{X}$ satisfies $\hat{X}^2 = A + \Delta A$, where $\|\Delta A\|_F \approx O(\epsilon)n\left\|\hat{X}\right\|_F^2$, where ϵ is ***machine precision***.

8 Further Comments

The cost of the algorithm is $n^3/3$ floating-point operations; see Algorithm 6.7 of Higham (2008). $O(n)$ of integer allocatable memory is required by the function.

If A is a full matrix, then nag_matop_real_gen_matrix_sqrt (f01en) should be used to compute the square root. If A has negative real eigenvalues then nag_matop_complex_gen_matrix_sqrt (f01fn) can be used to return a complex, non-principal square root.

If condition number and residual bound estimates are required, then nag_matop_real_gen_matrix_cond_sqrt (f01jd) 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

$$A = \begin{pmatrix} 6 & 4 & -5 & 15 \\ 8 & 6 & -3 & 10 \\ 0 & 0 & 3 & -4 \\ 0 & 0 & 4 & 3 \end{pmatrix}.$$

9.1 Program Text

```
function f01ep_example

fprintf('f01ep example results\n\n');

% Principal square root of matrix A

a = [ 6  4 -5 15;
      8  6 -3 10;
      0  0  3 -4;
      0  0  4  3];

[as, ifail] = f01ep(a);

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

9.2 Program Results

```
f01ep example results

Square root of A:
    2.0000    1.0000   -2.0000    3.0000
    2.0000    2.0000    0.0000    1.0000
         0         0    2.0000   -1.0000
         0         0    1.0000    2.0000
```