NAG Library Function Document

nag_matop_real_gen_matrix_cond_sqrt (f01jdc)

1 Purpose

nag_matop_real_gen_matrix_cond_sqrt (f01jdc) computes an estimate of the relative condition number $\kappa_{A^{1/2}}$ and a bound on the relative residual, in the Frobenius norm, for the square root of a real n by n matrix A. The principal square root, $A^{1/2}$, of A is also returned.

2 Specification

```
#include <nag.h>
#include <nagf01.h>
```

```
void nag_matop_real_gen_matrix_cond_sqrt (Integer n, double a[],
     Integer pda, double *alpha, double *condsa, NagError *fail)
```

3 Description

For a matrix with no eigenvalues on the closed negative real line, the principal matrix square root, $A^{1/2}$, of A is the unique square root with eigenvalues in the right half-plane.

The Fréchet derivative of a matrix function $A^{1/2}$ in the direction of the matrix E is the linear function mapping E to $L(A,E)$ such that

$$(A+E)^{1/2} - A^{1/2} - L(A,E) = o(\|A\|).$$

The absolute condition number is given by the norm of the Fréchet derivative which is defined by

$$\|L(A)\| := \max_{E\neq 0}\frac{\|L(A,E)\|}{\|E\|}.$$

The Fréchet derivative is linear in E and can therefore be written as

$$\mathrm{vec}(L(A,E)) = K(A)\mathrm{vec}(E),$$

where the vec operator stacks the columns of a matrix into one vector, so that $K(A)$ is $n^2 \times n^2$.

nag_matop_real_gen_matrix_cond_sqrt (f01jdc) uses Algorithm 3.20 from Higham (2008) to compute an estimate γ such that $\gamma \leq \|K(X)\|_F$. The quantity of γ provides a good approximation to $\|L(A)\|_F$. The relative condition number, $\kappa_{A^{1/2}}$, is then computed via

$$\kappa_{A^{1/2}} = \frac{\|L(A)\|_F\|A\|_F}{\|A^{1/2}\|_F}.$$

$\kappa_{A^{1/2}}$ is returned in the argument **condsa**.

$A^{1/2}$ is computed using the algorithm described in Higham (1987). This is a real arithmetic version of the algorithm of Björck and Hammarling (1983). In addition, a blocking scheme described in Deadman *et al.* (2013) is used.

The computed quantity α is a measure of the stability of the relative residual (see Section 7). It is computed via

$$\alpha = \frac{\|A^{1/2}\|_F^2}{\|A\|_F}.$$

4 References

Björck Å 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. Öster, 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 Arguments

1: **n** – Integer *Input*

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

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

2: **a**$[dim]$ – double *Input/Output*

Note: the dimension, *dim*, of the array **a** must be at least $\mathbf{pda} \times \mathbf{n}$.

The (i, j)th element of the matrix A is stored in $\mathbf{a}[(j-1) \times \mathbf{pda} + i - 1]$.

On entry: the n by n matrix A.

On exit: contains, if **fail.code** = NE_NOERROR, the n by n principal matrix square root, $A^{1/2}$. Alternatively, if **fail.code** = NE_EIGENVALUES, contains an n by n non-principal square root of A.

3: **pda** – Integer *Input*

On entry: the stride separating matrix row elements in the array **a**.

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

4: **alpha** – double * *Output*

On exit: an estimate of the stability of the relative residual for the computed principal (if **fail.code** = NE_NOERROR) or non-principal (if **fail.code** = NE_EIGENVALUES) matrix square root, α.

5: **condsa** – double * *Output*

On exit: an estimate of the relative condition number, in the Frobenius norm, of the principal (if **fail.code** = NE_NOERROR) or non-principal (if **fail.code** = NE_EIGENVALUES) matrix square root at A, $\kappa_{A^{1/2}}$.

6: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_ALG_FAIL

An error occurred when computing the condition number. The matrix square root was still returned but you should use nag_matop_real_gen_matrix_sqrt (f01enc) to check if it is the principal matrix square root.

An error occurred when computing the matrix square root. Consequently, **alpha** and **condsa** could not be computed. It is likely that the function was called incorrectly.

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM

On entry, argument $\langle value \rangle$ had an illegal value.

NE_EIGENVALUES

A has a semisimple vanishing eigenvalue. A non-principal square root was returned.

NE_INT

On entry, $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{n} \geq 0$.

NE_INT_2

On entry, $\mathbf{pda} = \langle value \rangle$ and $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{pda} \geq \mathbf{n}$.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in the Essential Introduction for further information.

NE_NEGATIVE_EIGVAL

A has a negative real eigenvalue. The principal square root is not defined.
nag_matop_complex_gen_matrix_cond_sqrt (f01kdc) can be used to return a complex, non-principal square root.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

NE_SINGULAR

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

7 Accuracy

If the computed square root is $\tilde{X}$, then the relative residual

$$\frac{\|A - \tilde{X}^2\|_F}{\|A\|_F},$$

is bounded approximately by $n\alpha\epsilon$, where ϵ is ***machine precision***. The relative error in $\tilde{X}$ is bounded approximately by $n\alpha\kappa_{A^{1/2}}\epsilon$.

8 Parallelism and Performance

nag_matop_real_gen_matrix_cond_sqrt (f01jdc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_matop_real_gen_matrix_cond_sqrt (f01jdc) 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 function. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

Approximately $3 \times n^2$ of real allocatable memory is required by the function.

The cost of computing the matrix square root is $85n^3/3$ floating-point operations. The cost of computing the condition number depends on how fast the algorithm converges. It typically takes over twice as long as computing the matrix square root.

If condition estimates are not required then it is more efficient to use nag_matop_real_gen_matrix_sqrt (f01enc) to obtain the matrix square root alone. Condition estimates for the square root of a complex matrix can be obtained via nag_matop_complex_gen_matrix_cond_sqrt (f01kdc).

10 Example

This example estimates the matrix square root and condition number of the matrix

$$A = \begin{pmatrix} -5 & 2 & -1 & 1 \\ -2 & -3 & 19 & 27 \\ -9 & 0 & 15 & 24 \\ 7 & 8 & 11 & 16 \end{pmatrix}.$$

10.1 Program Text

```
/* nag_matop_real_gen_matrix_cond_sqrt (f01jdc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 24, 2013.
 */
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf01.h>
#include <nagx04.h>

#define A(I,J) a[J*pda + I]

int main(void)
{
  /* Scalars */
  Integer        exit_status = 0;
  Integer        i, j, n, pda;
  double         alpha, condsa;
  /* Arrays */
  double         *a = 0;
  /* Nag Types */
  Nag_OrderType  order = Nag_ColMajor;
  NagError       fail;

  INIT_FAIL(fail);

  /* Output preamble */
  printf("nag_matop_real_gen_matrix_cond_sqrt (f01jdc) ");
  printf("Example Program Results\n\n");
```

```
  fflush(stdout);

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  /* Read in the problem size */
#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%*[^\n]", &n);
#else
  scanf("%"NAG_IFMT"%*[^\n]", &n);
#endif

  pda = n;
  if (!(a = NAG_ALLOC(pda*n, double))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }

  /* Read in the matrix A from data file */
  for (i = 0; i < n; i++)
#ifdef _WIN32
    for (j = 0; j < n; j++) scanf_s("%lf", &A(i, j));
#else
    for (j = 0; j < n; j++) scanf("%lf", &A(i, j));
#endif
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  /* Find matrix square root, condition number and residual bound using
   * nag_matop_real_gen_matrix_cond_sqrt (f01jdc)
   * Condition number for the square root of a real matrix
   */
  nag_matop_real_gen_matrix_cond_sqrt (n, a, pda, &alpha, &condsa, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_matop_real_gen_matrix_cond_sqrt (f01jdc)\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  /* Print matrix sqrt(A) using nag_gen_real_mat_print (x04cac)
   * Print real general matrix (easy-to-use)
   */
  nag_gen_real_mat_print (order, Nag_GeneralMatrix, Nag_NonUnitDiag,
                          n, n, a, pda, "sqrt(A)", NULL, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_gen_real_mat_print (x04cac)\n%s\n", fail.message);
    exit_status = 2;
    goto END;
  }

  /* Print condition number estimates */
  printf("Estimated relative condition number is: %7.2f\n", condsa);
  printf("Condition number for the relative residual is: %7.2f\n",alpha);

 END:
  NAG_FREE(a);
  return exit_status;
}
```

10.2 Program Data

```
nag_matop_real_gen_matrix_cond_sqrt (f01jdc) Example Program Data

4                              :Value of n

-5.0     2.0     -1.0     1.0
-2.0    -3.0     19.0    27.0
-9.0     0.0     15.0    24.0
 7.0     8.0     11.0    16.0  :End of matrix a
```

10.3 Program Results

```
nag_matop_real_gen_matrix_cond_sqrt (f01jdc) Example Program Results

 sqrt(A)
           1          2          3          4
 1      1.0000     2.0000    -1.0000    -1.0000
 2     -3.0000     1.0000     2.0000     4.0000
 3     -2.0000     3.0000     1.0000     2.0000
 4      2.0000    -1.0000     3.0000     4.0000
Estimated relative condition number is:   77.10
Condition number for the relative residual is:    1.70
```