NAG Library Function Document

nag_matop_real_gen_matrix_frcht_log (f01jkc)

1   Purpose

nag_matop_real_gen_matrix_frcht_log (f01jkc) computes the Fréchet derivative $L(A, E)$ of the matrix logarithm of the real n by n matrix A applied to the real n by n matrix E. The principal matrix logarithm $\log(A)$ is also returned.

2   Specification

```
#include <nag.h>
#include <nagf01.h>
```
```
void nag_matop_real_gen_matrix_frcht_log (Integer n, double a[],
    Integer pda, double e[], Integer pde, NagError *fail)
```

3   Description

For a matrix with no eigenvalues on the closed negative real line, the principal matrix logarithm $\log(A)$ is the unique logarithm whose spectrum lies in the strip $\{z : -\pi < \mathrm{Im}(z) < \pi\}$.

The Fréchet derivative of the matrix logarithm of A is the unique linear mapping $E \mapsto L(A, E)$ such that for any matrix E

$$\log(A + E) - \log(A) - L(A, E) = o(\|E\|).$$

The derivative describes the first order effect of perturbations in A on the logarithm $\log(A)$.

nag_matop_real_gen_matrix_frcht_log (f01jkc) uses the algorithm of Al–Mohy *et al.* (2012) to compute $\log(A)$ and $L(A, E)$. The principal matrix logarithm $\log(A)$ is computed using a Schur decomposition, a Padé approximant and the inverse scaling and squaring method. The Padé approximant is then differentiated in order to obtain the Fréchet derivative $L(A, E)$.

4   References

Al–Mohy A H and Higham N J (2011) Improved inverse scaling and squaring algorithms for the matrix logarithm *SIAM J. Sci. Comput.* **34(4)** C152–C169

Al–Mohy A H, Higham N J and Relton S D (2012) Computing the Fréchet derivative of the matrix logarithm and estimating the condition number *MIMS EPrint* **2012.72**

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: **n** ≥ 0.

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

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

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

   On entry: the n by n matrix A.

   On exit: the n by n principal matrix logarithm, $\log(A)$.

3:    **pda** – Integer   *Input*

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

   Constraint: **pda** $\geq$ **n**.

4:    **e**[*dim*] – double   *Input/Output*

   Note: the dimension, *dim*, of the array **e** must be at least **pde** $\times$ **n**.

   The (i,j)th element of the matrix E is stored in **e**$[(j-1) \times$ **pde** $+ i - 1]$.

   On entry: the n by n matrix E

   On exit: the Fréchet derivative $L(A, E)$

5:    **pde** – Integer   *Input*

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

   Constraint: **pde** $\geq$ **n**.

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

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

6   Error Indicators and Warnings

NE_ALLOC_FAIL

   Dynamic memory allocation failed.

NE_BAD_PARAM

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

NE_INT

   On entry, **n** $= \langle value \rangle$.
   Constraint: **n** ≥ 0.

NE_INT_2

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

   On entry, **pde** $= \langle value \rangle$ and **n** $= \langle value \rangle$.
   Constraint: **pde** $\geq$ **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.

NE_NEGATIVE_EIGVAL

   A has eigenvalues on the negative real line. The principal logarithm is not defined in this case; nag_matop_complex_gen_matrix_frcht_log (f01kkc) can be used to return a complex, non-principal log.

NE_SINGULAR

   A is singular so the logarithm cannot be computed.

NW_SOME_PRECISION_LOSS

$\log(A)$ has been computed using an IEEE double precision Padé approximant, although the arithmetic precision is higher than IEEE double precision.

7   Accuracy

For a normal matrix A (for which $A^{\mathrm{T}}A = AA^{\mathrm{T}}$), the Schur decomposition is diagonal and the computation of the matrix logarithm reduces to evaluating the logarithm of the eigenvalues of A and then constructing $\log(A)$ using the Schur vectors. This should give a very accurate result. In general, however, no error bounds are available for the algorithm. The sensitivity of the computation of $\log(A)$ and $L(A, E)$ is worst when A has an eigenvalue of very small modulus or has a complex conjugate pair of eigenvalues lying close to the negative real axis. See Al–Mohy and Higham (2011), Al–Mohy *et al.* (2012) and Section 11.2 of Higham (2008) for details and further discussion.

8   Parallelism and Performance

nag_matop_real_gen_matrix_frcht_log (f01jkc) is threaded by NAG for parallel execution in multi-threaded implementations of the NAG Library.

nag_matop_real_gen_matrix_frcht_log (f01jkc) 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 Users' Note for your implementation for any additional implementation-specific information.

9   Further Comments

The cost of the algorithm is $O(n^3)$ floating-point operations. The real allocatable memory required is approximately $5n^2$; see Al–Mohy *et al.* (2012) for further details.

If the matrix logarithm alone is required, without the Fréchet derivative, then nag_matop_real_gen_matrix_log (f01ejc) should be used. If the condition number of the matrix logarithm is required then nag_matop_real_gen_matrix_cond_log (f01jjc) should be used. If A has negative real eigenvalues then nag_matop_complex_gen_matrix_frcht_log (f01kkc) can be used to return a complex, non-principal matrix logarithm and its Fréchet derivative $L(A, E)$.

10   Example

This example finds the principal matrix logarithm $\log(A)$ and the Fréchet derivative $L(A, E)$, where

$$A = \begin{pmatrix} 4 & 2 & 0 & 2 \\ 3 & 3 & 1 & 1 \\ 3 & 2 & 1 & 0 \\ 3 & 3 & 1 & 2 \end{pmatrix} \quad \text{and} \quad E = \begin{pmatrix} 1 & 2 & 2 & 2 \\ 0 & 0 & 3 & 1 \\ 1 & 2 & 1 & 2 \\ 1 & 3 & 1 & 1 \end{pmatrix}.$$

10.1 Program Text

```
/* nag_matop_real_gen_matrix_frcht_log (f01jkc) Example Program.
 *
 * Copyright 2013 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]
#define E(I,J) e[J*pde + I]
```

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

  INIT_FAIL(fail);

  /* Output preamble */
  printf("nag_matop_real_gen_matrix_frcht_log (f01jkc) ");
  printf("Example Program Results\n\n");

  /* Skip heading in data file */
  scanf("%*[^\n] ");

  /* Read in the problem size */
  scanf("%ld%*[^\n] ", &n);

  pda = n;
  if (!(a = NAG_ALLOC(pda*n, double))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  pde = n;
  if (!(e = NAG_ALLOC(pde*n, double))) {
    printf("Allocation failure\n");
    exit_status = -2;
    goto END;
  }

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

  /* Read in the matrix E from data file */
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      scanf("%lf", &E(i, j));
  scanf("%*[^\n] ");

  /* Find log(A) and L(A,E) using
   * nag_matop_real_gen_matrix_frcht_log (f01jkc)
   * Frechet derivative of real matrix logarithm
   */
  nag_matop_real_gen_matrix_frcht_log(n, a, pda, e, pde, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_matop_real_gen_matrix_frcht_log (f01jkc)\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  /* Print matrix log(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, "log(A)", 0, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_gen_real_mat_print (x04cac)\n%s\n", fail.message);
    exit_status = 2;
```

```
  }

  /* Print matrix L(A,E) 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,
                         e, pde, "L(A,E)", 0, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_gen_real_mat_print (x04cac)\n%s\n", fail.message);
    exit_status = 3;
  }

 END:
  NAG_FREE(a);
  NAG_FREE(e);

  return exit_status;
}
```

10.2 Program Data

```
nag_matop_real_gen_matrix_frcht_log (f01jkc) Example Program Data

  4                         :Value of n

  4.0    2.0    0.0    2.0
  3.0    3.0    1.0    1.0
  3.0    2.0    1.0    0.0
  3.0    3.0    1.0    2.0   :End of matrix a

  1.0    2.0    2.0    2.0
  0.0    0.0    3.0    1.0
  1.0    2.0    1.0    2.0
  1.0    3.0    1.0    1.0   :End of matrix e
```

10.3 Program Results

```
nag_matop_real_gen_matrix_frcht_log (f01jkc) Example Program Results

 log(A)
            1         2         3         4
 1      1.1165    0.5296   -0.4079    0.6962
 2      0.6996    0.2025    0.8192    0.4745
 3      1.3114    1.5867   -0.1433   -1.1720
 4      0.5272    1.2856    0.4055    0.2106
 L(A,E)
            1         2         3         4
 1     -0.1211    0.1974    0.1463    0.8268
 2     -1.2615   -4.1260    3.4035    2.4651
 3      1.2387    5.7968   -3.6489   -2.7203
 4      0.6231    3.7059   -1.9334   -1.8540
```