NAG Library Function Document

nag_matop_complex_gen_matrix_frcht_log (f01kkc)

1 Purpose

nag_matop_complex_gen_matrix_frcht_log (f01kkc) computes the Fréchet derivative $L(A,E)$ of the matrix logarithm of the complex n by n matrix A applied to the complex 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_complex_gen_matrix_frcht_log (Integer n, Complex a[],
     Integer pda, Complex 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_complex_gen_matrix_frcht_log (f01kkc) 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 $L(A,E)$. If A is nonsingular but has negative real eigenvalues, the principal logarithm is not defined, but nag_matop_complex_gen_matrix_frcht_log (f01kkc) will return a non-principal logarithm and Fréchet derivative.

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

2: $\mathbf{a}[\mathit{dim}]$ – Complex *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: the n by n principal matrix logarithm, $\log(A)$. Alterntively, if **fail.code** = NE_NEGATIVE_EIGVAL, a non-principal logarithm is returned.

3: **pda** – Integer *Input*

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

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

4: **e**[*dim*] – Complex *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: with **fail.code** = NE_NOERROR, NE_NEGATIVE_EIGVAL or NW_SOME_PRECISION_LOSS, 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.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 0.

NE_INT_2

On entry, **pda** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: **pda** $\geq$ **n**.

On entry, **pde** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
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.

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 eigenvalues on the negative real line. The principal logarithm is not defined in this case, so a non-principal logarithm was returned.

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 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 $A^{\mathrm{H}}A = AA^{\mathrm{H}}$), 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_complex_gen_matrix_frcht_log (f01kkc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_matop_complex_gen_matrix_frcht_log (f01kkc) 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

The cost of the algorithm is $O(n^3)$ floating-point operations. The complex 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_complex_gen_matrix_log (f01fjc) should be used. If the condition number of the matrix logarithm is required then nag_matop_complex_gen_matrix_cond_log (f01kjc) should be used. The real analogue of this function is nag_matop_real_gen_matrix_frcht_log (f01jkc).

10 Example

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

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

10.1 Program Text

```
/* nag_matop_complex_gen_matrix_frcht_log (f01kkc) 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]
#define E(I,J) e[J*pde + I]

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

  INIT_FAIL(fail);

  printf("nag_matop_complex_gen_matrix_frcht_log (f01kkc) ");
  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, Complex))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  pde = n;
  if (!(e = NAG_ALLOC(pde*n, Complex))) {
    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++)
#ifdef _WIN32
      scanf_s(" ( %lf , %lf ) ", &A(i, j).re, &A(i, j).im);
#else
      scanf(" ( %lf , %lf ) ", &A(i, j).re, &A(i, j).im);
#endif
#ifdef _WIN32
```

```
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif

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

  /* Find log(A) and L(A,E) using
   * nag_matop_complex_gen_matrix_frcht_log (f01kkc)
   * Frechet derivative of complex matrix logarithm
   */
  nag_matop_complex_gen_matrix_frcht_log(n, a, pda, e, pde, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_matop_complex_gen_matrix_frcht_log (f01kkc)\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  /* Print matrix log(A) using nag_gen_cmplx_mat_print (x04dac)
   * Print complex general matrix (easy-to-use)
   */
  nag_gen_complx_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag,
                           n, n, a, pda, "log(A)", NULL, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_gen_cmplx_mat_print (x04dac)\n%s\n", fail.message);
    exit_status = 2;
    goto END;
  }

  /* Print matrix L(A,E) using nag_gen_cmplx_mat_print (x04dac)
   * Print complex general matrix (easy-to-use)
   */
  nag_gen_complx_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag,
                           n, n, e, pde, "L(A,E)", NULL, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_gen_cmplx_mat_print (x04dac)\n%s\n", fail.message);
    exit_status = 3;
    goto END;
  }

 END:

  NAG_FREE(a);
  NAG_FREE(e);

  return exit_status;
}
```

10.2 Program Data

```
nag_matop_complex_gen_matrix_frcht_log (f01kkc) Example Program Data

  4                                       :Value of n

  (1.0,4.0)  (0.0,3.0)  (0.0,1.0)  (2.0,0.0)
  (0.0,2.0)  (3.0,0.0)  (1.0,0.0)  (1.0,1.0)
  (0.0,1.0)  (2.0,1.0)  (2.0,0.0)  (0.0,1.0)
  (1.0,2.0)  (3.0,2.0)  (1.0,2.0)  (3.0,1.0)  :End of matrix a

  (1.0,0.0)  (1.0,2.0)  (2.0,0.0)  (2.0,1.0)
  (1.0,3.0)  (0.0,1.0)  (1.0,0.0)  (0.0,0.0)
  (0.0,2.0)  (4.0,1.0)  (1.0,0.0)  (1.0,0.0)
  (1.0,0.0)  (2.0,2.0)  (0.0,3.0)  (1.0,0.0)  :End of matrix e
```

10.3 Program Results

```
nag_matop_complex_gen_matrix_frcht_log (f01kkc) Example Program Results

 log(A)
               1           2           3           4
 1        1.4188      0.2758     -0.2240      0.4528
          1.2438      1.0040      0.0826     -0.5887

 2        0.2299      1.0702      0.5292      0.1976
          0.4825     -0.3306     -0.0422      0.1532

 3        0.1328      0.9235      0.6051     -0.1211
         -0.0462      0.3060     -0.0973      0.2966

 4        0.4704      1.0779      0.2724      0.9612
         -0.0891      0.0538      0.7627      0.2680

 L(A,E)
               1           2           3           4
 1        0.1620     -0.0593     -0.1543      0.5534
         -0.6532      0.8434     -1.3537      0.0869

 2        0.6673      0.0637      0.3421     -0.4639
          0.7351     -0.0911      0.1136     -0.3399

 3       -0.2500      1.4898     -0.1547      0.3319
         -0.0433      0.6186     -0.0495     -0.3078

 4       -0.4004      0.5834     -0.5153      0.4407
         -0.5893     -0.5926      1.4107      0.1236
```