NAG Library Function Document

nag_matop_complex_gen_matrix_frcht_pow (f01kfc)

1 Purpose

nag_matop_complex_gen_matrix_frcht_pow (f01kfc) computes the Fréchet derivative $L(A,E)$ of the pth power (where p is real) of the complex n by n matrix A applied to the complex n by n matrix E. The principal matrix power A^p is also returned.

2 Specification

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

3 Description

For a matrix A with no eigenvalues on the closed negative real line, A^p ($p \in \mathbb{R}$) can be defined as

$$A^p = \exp(p\log(A))$$

where $\log(A)$ is the principal logarithm of A (the unique logarithm whose spectrum lies in the strip $\{z : -\pi < \mathrm{Im}(z) < \pi\}$). If A is nonsingular but has negative real eigenvalues, the principal logarithm is not defined, but a non-principal pth power can be defined by using a non-principal logarithm.

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

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

The derivative describes the first-order effect of perturbations in A on the matrix power A^p.

nag_matop_complex_gen_matrix_frcht_pow (f01kfc) uses the algorithms of Higham and Lin (2011) and Higham and Lin (2013) to compute A^p and $L(A,E)$. The real number p is expressed as $p = q + r$ where $q \in (-1,1)$ and $r \in \mathbb{Z}$. Then $A^p = A^q A^r$. The integer power A^r is found using a combination of binary powering and, if necessary, matrix inversion. The fractional power A^q is computed using a Schur decomposition, a Padé approximant and the scaling and squaring method. The Padé approximant is differentiated in order to obtain the Fréchet derivative of A^q and $L(A,E)$ is then computed using a combination of the chain rule and the product rule for Fréchet derivatives.

4 References

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

Higham N J and Lin L (2011) A Schur–Padé algorithm for fractional powers of a matrix *SIAM J. Matrix Anal. Appl.* **32(3)** 1056–1078

Higham N J and Lin L (2013) An improved Schur–Padé algorithm for fractional powers of a matrix and their Fréchet derivatives *SIAM J. Matrix Anal. Appl.* **34(3)** 1341–1360

5 Arguments

1: **n** – Integer *Input*

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

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

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

Note: the dimension, *dim*, of the array **a** must be at least **pda** × **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 pth power, A^p. Alternatively if **fail.code** = NE_NEGATIVE_EIGVAL, a non-principal pth power 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** × **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: **p** – double *Input*

On entry: the required power of A.

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

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 2.3.1.2 in How to Use the NAG Library and its Documentation 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 2.7.6 in How to Use the NAG Library and its Documentation for further information.

NE_NEGATIVE_EIGVAL

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

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

NE_SINGULAR

A is singular so the pth power cannot be computed.

NW_SOME_PRECISION_LOSS

A^p 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 fractional part of the matrix power reduces to evaluating powers of the eigenvalues of A and then constructing A^p using the Schur vectors. This should give a very accurate result. In general, however, no error bounds are available for the algorithm. See Higham and Lin (2011) and Higham and Lin (2013) for details and further discussion.

If the condition number of the matrix power is required then nag_matop_complex_gen_matrix_cond_pow (f01kec) should be used.

8 Parallelism and Performance

nag_matop_complex_gen_matrix_frcht_pow (f01kfc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_matop_complex_gen_matrix_frcht_pow (f01kfc) 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 complex allocatable memory required by the algorithm is approximately $6 \times n^2$.

The cost of the algorithm is $O\left(n^3\right)$ floating-point operations; see Higham and Lin (2011) and Higham and Lin (2013).

If the matrix pth power alone is required, without the Fréchet derivative, then nag_matop_complex_gen_matrix_pow (f01fqc) should be used. If the condition number of the matrix power is required then nag_matop_complex_gen_matrix_cond_pow (f01kec) should be used. The real analogue of this function is nag_matop_real_gen_matrix_frcht_pow (f01jfc).

10 Example

This example finds A^p and the Fréchet derivative of the matrix power $L(A,E)$, where $p = 0.2$,

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

10.1 Program Text

```
/* nag_matop_complex_gen_matrix_frcht_pow (f01kfc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
#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;
  double p;
  /* 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_pow (f01kfc) ");
  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 and the required power */
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "", &n);
#else
  scanf("%" NAG_IFMT "", &n);
#endif
#ifdef _WIN32
  scanf_s("%lf", &p);
#else
  scanf("%lf", &p);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\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 A^p and L(A,E) using
   * nag_matop_complex_gen_matrix_frcht_pow (f01kfc)
   * Frechet derivative of complex matrix power
   */
  nag_matop_complex_gen_matrix_frcht_pow(n, a, pda, e, pde, p, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_matop_complex_gen_matrix_frcht_pow (f01kfc)\n%s\n",
            fail.message);
    exit_status = 1;
    goto END;
  }

  /* Print matrix A^p 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, "A^p", 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_pow (f01kfc) Example Program Data

4     0.2                                        :Values of n and p

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

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

10.3 Program Results

```
nag_matop_complex_gen_matrix_frcht_pow (f01kfc) Example Program Results

 A^p
             1           2           3           4
 1      1.2029      0.0810      0.2374     -0.0520
       -0.0424      0.0428     -0.1718      0.0976

 2      0.1311      1.1054     -0.0757      0.2308
       -0.0378      0.1091      0.0066      0.1373

 3     -0.0305      0.4878      1.0822     -0.1050
       -0.1948      0.2846      0.2620      0.3131

 4      0.3401     -0.3005      0.1838      1.2347
        0.1792     -0.0857     -0.0261     -0.1571

 L(A,E)
           1         2         3         4
 1   0.0980  -0.0980   0.0410   0.0136
    -0.0926   0.2759  -0.2629   0.1853

 2  -0.0644  -0.2093   0.4315   0.1337
     0.3359  -0.3976   0.0395  -0.0976

 3   0.1912   0.2279  -0.0963  -0.0925
     0.0032   0.3308   0.1146  -0.3254

 4  -0.0907  -0.0153   0.1299   0.2238
     0.1255  -0.4022   0.0694   0.1179
```