# NAG Library Function Document

# nag_matop_real_gen_matrix_frcht_pow (f01jfc)

## 1 Purpose

nag_matop_real_gen_matrix_frcht_pow (f01jfc) computes the Fréchet derivative $L(A,E)$ of the $p$th power (where $p$ is real) of the real $n$ by $n$ matrix $A$ applied to the real $n$ by $n$ matrix $E$. The principal matrix power $A^p$ is also returned.

## 2 Specification

```
#include <nag.h>
#include <nagf01.h>
void nag_matop_real_gen_matrix_frcht_pow (Integer n, double a[],
     Integer pda, double 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 $\{z : -\pi < \mathrm{Im}(z) < \pi\}$).

The Fréchet derivative of the matrix $p$th 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_real_gen_matrix_frcht_pow (f01jfc) 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 *MIMS Eprint 2013.1* Manchester Institute for Mathematical Sciences, School of Mathematics, University of Manchester http://eprints.ma.man.ac.uk/

## 5 Arguments

1: **n** – Integer *Input*

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

*Constraint*: **n** $\geq 0$.

2: **a**[$dim$] – double *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 $p$th power, $A^p$.

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** × **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 3.6 in the Essential Introduction).

## 6 Error Indicators and Warnings

**NE_ALLOC_FAIL**

Dynamic memory allocation failed.

**NE_BAD_PARAM**

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

**NE_INT**

On entry, **n** = ⟨*value*⟩.
Constraint: **n** $\geq 0$.

**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.

**NE_NEGATIVE_EIGVAL**

$A$ has eigenvalues on the negative real line. The principal $p$th power is not defined in this case; nag_matop_complex_gen_matrix_frcht_pow (f01kfc) can be used to find a complex, non-principal $p$th power.

**NE_SINGULAR**

$A$ is singular so the $p$th 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 $A^{\mathrm{T}}A = AA^{\mathrm{T}}$), 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_real_gen_matrix_cond_pow (f01jec) should be used.

## 8 Parallelism and Performance

nag_matop_real_gen_matrix_frcht_pow (f01jfc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_matop_real_gen_matrix_frcht_pow (f01jfc) 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 real allocatable memory required by the algorithm is approximately $6 \times n^2$.

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

If the matrix $p$th power alone is required, without the Fréchet derivative, then nag_matop_real_gen_matrix_pow (f01eqc) should be used. If the condition number of the matrix power is required then nag_matop_real_gen_matrix_cond_pow (f01jec) should be used. If $A$ has negative real eigenvalues then nag_matop_complex_gen_matrix_frcht_pow (f01kfc) can be used to return a complex, non-principal $p$th power and its Fréchet derivative $L(A, E)$.

## 10 Example

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

$$A = \begin{pmatrix} 3 & 3 & 2 & 1 \\ 3 & 1 & 0 & 2 \\ 1 & 1 & 4 & 3 \\ 3 & 0 & 3 & 1 \end{pmatrix} \quad \text{and} \quad E = \begin{pmatrix} 1 & 0 & 2 & 1 \\ 0 & 4 & 5 & 2 \\ 1 & 0 & 0 & 0 \\ 2 & 3 & 3 & 0 \end{pmatrix}.$$

## 10.1 Program Text

```
/* nag_matop_real_gen_matrix_frcht_pow (f01jfc) 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;
  double        p;
  /* 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_pow (f01jfc) ");
  printf("Example Program Results\n\n");

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

  /* Read in the problem size and required power */
  scanf("%ld", &n);
  scanf("%lf", &p);
  scanf("%*[^\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 A^p and L(A,E) using
   * nag_matop_real_gen_matrix_frcht_pow (f01jfc)
   * Frechet derivative of real matrix power
   */
```

```
  nag_matop_real_gen_matrix_frcht_pow(n, a, pda, e, pde, p, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_matop_real_gen_matrix_frcht_pow (f01jfc)\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  /* Print matrix A^p 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, "A^p", 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_pow (f01jfc) Example Program Data

4     0.2              :Values of n and p

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

1.0   0.0   2.0   1.0
0.0   4.0   5.0   2.0
1.0   0.0   0.0   0.0
2.0   3.0   3.0   0.0  :End of matrix e
```

## 10.3 Program Results

```
nag_matop_real_gen_matrix_frcht_pow (f01jfc) Example Program Results

 A^p
            1          2          3          4
 1      1.2446     0.2375     0.2172    -0.1359
 2      0.0925     1.1239    -0.1453     0.3731
 3     -0.0769     0.1972     1.3131     0.1837
 4      0.3985    -0.2902     0.1085     1.1560
 L(A,E)
          1       2       3       4
 1   0.2189 -0.2004  0.0509  0.0290
 2  -0.3177  0.4143  0.3044  0.0760
 3  -0.0033 -0.1335 -0.2789  0.2699
 4   0.1972  0.3333  0.5379 -0.5228
```