

## NAG Library Function Document

### nag\_matop\_complex\_gen\_matrix\_cond\_pow (f01kec)

#### 1 Purpose

nag\_matop\_complex\_gen\_matrix\_cond\_pow (f01kec) computes an estimate of the relative condition number  $\kappa_{A^p}$  of the  $p$ th power (where  $p$  is real) of a complex  $n$  by  $n$  matrix  $A$ , in the 1-norm. The principal matrix power  $A^p$  is also returned.

#### 2 Specification

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

void nag_matop_complex_gen_matrix_cond_pow (Integer n, Complex a[],
      Integer pda, double p, double *condpa, 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  $\{z : -\pi < \text{Im}(z) < \pi\}$ ).

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  $A^p$ .

The relative condition number of the matrix  $p$ th power can be defined by

$$\kappa_{A^p} = \frac{\|L(A)\| \|A\|}{\|A^p\|},$$

where  $\|L(A)\|$  is the norm of the Fréchet derivative of the matrix power at  $A$ .

nag\_matop\_complex\_gen\_matrix\_cond\_pow (f01kec) uses the algorithms of Higham and Lin (2011) and Higham and Lin (2013) to compute  $\kappa_{A^p}$  and  $A^p$ . 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.

To obtain the estimate of  $\kappa_{A^p}$ , nag\_matop\_complex\_gen\_matrix\_cond\_pow (f01kec) first estimates  $\|L(A)\|$  by computing an estimate  $\gamma$  of a quantity  $K \in [n^{-1}\|L(A)\|_1, n\|L(A)\|_1]$ , such that  $\gamma \leq K$ . This requires multiple Fréchet derivatives to be computed. Fréchet derivatives of  $A^q$  are obtained by differentiating the Padé approximant. Fréchet derivatives of  $A^p$  are then computed using a combination of the chain rule and the product rule for Fréchet derivatives.

If  $A$  is nonsingular but has negative real eigenvalues nag\_matop\_complex\_gen\_matrix\_cond\_pow (f01kec) will return a non-principal matrix  $p$ th power and its condition number.

## 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:*  $n \geq 0$ .
- 2: **a**[*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 **a**[( $j - 1$ )  $\times$   $\mathbf{pda} + i - 1$ ].  
*On entry:* the  $n$  by  $n$  matrix  $A$ .  
*On exit:* the  $n$  by  $n$  principal matrix  $p$ th power,  $A^p$ , unless **fail.code** = NE\_NEGATIVE\_EIGVAL, in which case a non-principal  $p$ th power is returned.
- 3: **pda** – Integer *Input*  
*On entry:* the stride separating matrix row elements in the array **a**.  
*Constraint:*  $\mathbf{pda} \geq \mathbf{n}$ .
- 4: **p** – double *Input*  
*On entry:* the required power of  $A$ .
- 5: **condpa** – double \* *Output*  
*On exit:* if **fail.code** = NE\_NOERROR or NW\_SOME\_PRECISION\_LOSS, an estimate of the relative condition number of the matrix  $p$ th power,  $\kappa_{A^p}$ . Alternatively, if **fail.code** = NE\_RCOND, the absolute condition number of the matrix  $p$ th power.
- 6: **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  $\langle value \rangle$  had an illegal value.

**NE\_INT**

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

**NE\_INT\_2**

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

The relative condition number is infinite. The absolute condition number was returned instead.

**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**

`nag_matop_complex_gen_matrix_cond_pow` (f01kec) uses the norm estimation function `nag_linsys_complex_gen_norm_rcomm` (f04zdc) to produce an estimate  $\gamma$  of a quantity  $K \in [n^{-1}\|L(A)\|_1, n\|L(A)\|_1]$ , such that  $\gamma \leq K$ . For further details on the accuracy of norm estimation, see the documentation for `nag_linsys_complex_gen_norm_rcomm` (f04zdc).

For a normal matrix  $A$  (for which  $A^H A = A A^H$ ), 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.

**8 Parallelism and Performance**

`nag_matop_complex_gen_matrix_cond_pow` (f01kec) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

`nag_matop_complex_gen_matrix_cond_pow` (f01kec) 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 amount of complex allocatable memory required by the algorithm is typically of the order  $10 \times n^2$ .

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

If the matrix  $p$ th power alone is required, without an estimate of the condition number, then `nag_matop_complex_gen_matrix_pow` (f01fqc) should be used. If the Fréchet derivative of the matrix power is required then `nag_matop_complex_gen_matrix_frcht_pow` (f01kfc) should be used. The real analogue of this function is `nag_matop_real_gen_matrix_cond_pow` (f01jec).

## 10 Example

This example estimates the relative condition number of the matrix power  $A^p$ , where  $p = 0.4$  and

$$A = \begin{pmatrix} 1+2i & 3 & 2 & 1+3i \\ 1+i & 1 & 1 & 2+i \\ 1 & 2 & 1 & 2i \\ 3 & i & 2+i & 1 \end{pmatrix}.$$

### 10.1 Program Text

```
/* nag_matop_complex_gen_matrix_cond_pow (f01kec) 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]

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

    INIT_FAIL(fail);

    /* Output preamble */
    printf("nag_matop_complex_gen_matrix_cond_pow (f01kec) ");
    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("%i 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;
    }

    /* 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 , %lf ) ", &A(i, j).re, &A(i, j).im);
#else
        for (j = 0; j < n; j++)
            scanf(" ( %lf , %lf ) ", &A(i, j).re, &A(i, j).im);
#endif
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif

    /* Find the matrix pth power and condition number using
     * nag_matop_complex_gen_matrix_cond_pow (f01kec)
     * Condition number complex matrix power
     */
    nag_matop_complex_gen_matrix_cond_pow(n, a, pda, p, &condpa, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_matop_complex_gen_matrix_cond_pow (f01kec)\n%s\n",
            fail.message);
        exit_status = 1;
        goto END;
    }

    /* Print matrix A^p using nag_gen_complx_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_complx_mat_print (x04dac)\n%s\n",
            fail.message);
        exit_status = 2;
        goto END;
    }

    /* Print relative condition number estimate */
    printf("Estimated relative condition number is: %7.2f\n", condpa);

END:
    NAG_FREE(a);
    return exit_status;
}

```

## 10.2 Program Data

nag\_matop\_complex\_gen\_matrix\_cond\_pow (f01kec) Example Program Data

```

4      0.4                                     :Values of n and p

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

```

## 10.3 Program Results

nag\_matop\_complex\_gen\_matrix\_cond\_pow (f01kec) Example Program Results

```

A^p
      1      2      3      4
1    0.9742  0.8977  0.6389  0.0975
    0.5211 -0.1170 -0.3900  0.6205

2    0.1586  1.0176  0.0623  0.6431
    0.2763 -0.0250 -0.3471  0.2560

3    0.2589  0.5633  1.1470 -0.3771
   -0.5817  0.3969  0.4042  0.3113

4    0.8713 -0.5734  0.2816  1.3568
   -0.0270  0.0868  0.3739 -0.2709

```

Estimated relative condition number is: 6.86

---