

NAG Library Function Document

nag_matop_complex_gen_matrix_cond_std (f01kac)

1 Purpose

nag_matop_complex_gen_matrix_cond_std (f01kac) computes an estimate of the absolute condition number of a matrix function f of a complex n by n matrix A in the 1-norm, where f is either the exponential, logarithm, sine, cosine, hyperbolic sine (sinh) or hyperbolic cosine (cosh). The evaluation of the matrix function, $f(A)$, is also returned.

2 Specification

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

void nag_matop_complex_gen_matrix_cond_std (Nag_MatFunType fun, Integer n,
      Complex a[], Integer pda, double *conda, double *norma, double *normfa,
      NagError *fail)
```

3 Description

The absolute condition number of f at A , $\text{cond}_{\text{abs}}(f, A)$ is given by the norm of the Fréchet derivative of f , $L(A)$, which is defined by

$$\|L(X)\| := \max_{E \neq 0} \frac{\|L(X, E)\|}{\|E\|},$$

where $L(X, E)$ is the Fréchet derivative in the direction E . $L(X, E)$ is linear in E and can therefore be written as

$$\text{vec}(L(X, E)) = K(X)\text{vec}(E),$$

where the vec operator stacks the columns of a matrix into one vector, so that $K(X)$ is $n^2 \times n^2$. nag_matop_complex_gen_matrix_cond_std (f01kac) computes an estimate γ such that $\gamma \leq \|K(X)\|_1$, where $\|K(X)\|_1 \in [n^{-1}\|L(X)\|_1, n\|L(X)\|_1]$. The relative condition number can then be computed via

$$\text{cond}_{\text{rel}}(f, A) = \frac{\text{cond}_{\text{abs}}(f, A)\|A\|_1}{\|f(A)\|_1}.$$

The algorithm used to find γ is detailed in Section 3.4 of Higham (2008).

4 References

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

5 Arguments

- 1: **fun** – Nag_MatFunType *Input*
On entry: indicates which matrix function will be used.
- fun** = Nag_Exp
 The matrix exponential, e^A , will be used.
- fun** = Nag_Sin
 The matrix sine, $\sin(A)$, will be used.
- fun** = Nag_Cos
 The matrix cosine, $\cos(A)$, will be used.

fun = Nag_Sinh
The hyperbolic matrix sine, $\sinh(A)$, will be used.

fun = Nag_Cosh
The hyperbolic matrix cosine, $\cosh(A)$, will be used.

fun = Nag_Loga
The matrix logarithm, $\log(A)$, will be used.

Constraint: **fun** = Nag_Exp, Nag_Sin, Nag_Cos, Nag_Sinh, Nag_Cosh or Nag_Loga.

- 2: **n** – Integer *Input*
On entry: n , the order of the matrix A .
Constraint: $n \geq 0$.
- 3: **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 $\mathbf{a}[(j - 1) \times \mathbf{pda} + i - 1]$.
On entry: the n by n matrix A .
On exit: the n by n matrix, $f(A)$.
- 4: **pda** – Integer *Input*
On entry: the stride separating matrix row elements in the array **a**.
Constraint: $\mathbf{pda} \geq \mathbf{n}$.
- 5: **conda** – double * *Output*
On exit: an estimate of the absolute condition number of f at A .
- 6: **norma** – double * *Output*
On exit: the 1-norm of A .
- 7: **normfa** – double * *Output*
On exit: the 1-norm of $f(A)$.
- 8: **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, $\mathbf{n} = \langle value \rangle$.

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

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 internal error occurred when estimating the norm of the Fréchet derivative of f at A . Please contact NAG.

An internal error occurred when evaluating the matrix function $f(A)$. You can investigate further by calling `nag_matop_complex_gen_matrix_exp` (f01fcc), `nag_matop_complex_gen_matrix_log` (f01fjc) or `nag_matop_complex_gen_matrix_fun_std` (f01fkc) with the matrix A .

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_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.

7 Accuracy

`nag_matop_complex_gen_matrix_cond_std` (f01kac) uses the norm estimation function `nag_linsys_complex_gen_norm_rcomm` (f04zdc) to estimate a quantity γ , where $\gamma \leq \|K(X)\|_1$ and $\|K(X)\|_1 \in [n^{-1}\|L(X)\|_1, n\|L(X)\|_1]$. For further details on the accuracy of norm estimation, see the documentation for `nag_linsys_complex_gen_norm_rcomm` (f04zdc).

8 Parallelism and Performance

`nag_matop_complex_gen_matrix_cond_std` (f01kac) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library. In these implementations, this function may make calls to the user-supplied functions from within an OpenMP parallel region. Thus OpenMP pragmas within the user functions can only be used if you are compiling the user-supplied function and linking the executable in accordance with the instructions in the Users' Note for your implementation.

`nag_matop_complex_gen_matrix_cond_std` (f01kac) 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

Approximately $6n^2$ of complex allocatable memory is required by the routine, in addition to the memory used by the underlying matrix function routines `nag_matop_complex_gen_matrix_exp` (f01fcc), `nag_matop_complex_gen_matrix_log` (f01fjc) or `nag_matop_complex_gen_matrix_fun_std` (f01fkc).

`nag_matop_complex_gen_matrix_cond_std` (f01kac) returns the matrix function $f(A)$. This is computed using `nag_matop_complex_gen_matrix_exp` (f01fcc) if **fun** = Nag_Exp, `nag_matop_complex_gen_matrix_log` (f01fjc) if **fun** = Nag_Loga and `nag_matop_complex_gen_matrix_fun_std` (f01fkc) otherwise. If only $f(A)$ is required, without an estimate of the condition number, then it is far more efficient to use `nag_matop_complex_gen_matrix_exp` (f01fcc), `nag_matop_complex_gen_matrix_log` (f01fjc) or `nag_matop_complex_gen_matrix_fun_std` (f01fkc) directly.

`nag_matop_real_gen_matrix_cond_std` (f01jac) can be used to find the condition number of the exponential, logarithm, sine, cosine, sinh or cosh at a real matrix.

10 Example

This example estimates the absolute and relative condition numbers of the matrix sinh function for

$$A = \begin{pmatrix} 0.0 + 1.0i & -1.0 + 0.0i & 1.0 + 0.0i & 2.0 + 0.0i \\ 2.0 + 1.0i & 0.0 - 1.0i & 0.0 + 0.0i & 1.0 + 0.0i \\ 0.0 + 1.0i & 0.0 + 0.0i & 1.0 + 1.0i & 0.0 + 2.0i \\ 1.0 + 0.0i & 2.0 + 0.0i & -2.0 + 3.0i & 0.0 + 1.0i \end{pmatrix}.$$

10.1 Program Text

```

/* nag_matop_complex_gen_matrix_cond_std (f01kac) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf01.h>
#include <nagx02.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 conda, cond_rel, eps, norma, normfa;
    /* Arrays */
    Complex *a = 0;
    char nag_enum_arg[100];
    /* Nag Types */
    Nag_OrderType order = Nag_ColMajor;
    Nag_MatFunType fun;
    NagError fail;

    INIT_FAIL(fail);

    /* Output preamble */
    printf("nag_matop_complex_gen_matrix_cond_std (f01kac) ");
    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 function */
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "%99s%*[\n]", &n, nag_enum_arg,
            (unsigned)_countof(nag_enum_arg));
#else
    scanf("%" NAG_IFMT "%99s%*[\n]", &n, nag_enum_arg);
#endif

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

    /* nag_enum_name_to_value (x04nac)
     * Converts Nag enum member name to value
     */

```

```

fun = (Nag_MatFunType) nag_enum_name_to_value(nag_enum_arg);

/* 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
#endif
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif

/* Print matrix A 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", 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;
}

/* Find absolute condition number estimate using
 * nag_matop_complex_gen_matrix_cond_std (f01kac)
 * Condition number for the exponential, logarithm, sine, cosine,
 * sinh or cosh of a complex matrix
 */
nag_matop_complex_gen_matrix_cond_std(fun, n, a, pda, &conda,
                                       &norma, &normfa, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_matop_complex_gen_matrix_cond_std (f01kac)\n%s\n",
        fail.message);
    exit_status = 1;
    goto END;
}

/* Print absolute condition number estimate */
printf("\nF(A) = %s(A)\n", nag_enum_arg);
printf("Estimated absolute condition number is: %7.2f\n", conda);

/* nag_machine_precision (x02ajc) The machine precision */
eps = nag_machine_precision;

/* Find relative condition number estimate */
if (normfa > eps) {
    cond_rel = conda * norma / normfa;
    printf("Estimated relative condition number is: %7.2f\n", cond_rel);
}
else {
    printf("The estimated norm of f(A) is effectively zero");
    printf("and so the relative condition number is undefined.\n");
}

END:
    NAG_FREE(a);
    return exit_status;
}

```

10.2 Program Data

nag_matop_complex_gen_matrix_cond_std (f01kac) Example Program Data

```

4                Nag_Sinh                :Values of n and fun

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

```

10.3 Program Results

nag_matop_complex_gen_matrix_cond_std (f01kac) Example Program Results

```
A
      1      2      3      4
1    0.0000  -1.0000  1.0000  2.0000
    1.0000   0.0000  0.0000  0.0000

2    2.0000   0.0000  0.0000  1.0000
    1.0000  -1.0000  0.0000  0.0000

3    0.0000   0.0000  1.0000  0.0000
    1.0000   0.0000  1.0000  2.0000

4    1.0000   2.0000  -2.0000  0.0000
    0.0000   0.0000   3.0000  1.0000
```

F(A) = Nag_Sinh(A)

Estimated absolute condition number is: 7.33

Estimated relative condition number is: 4.94
