NAG Library Function Document

nag_matop_complex_gen_matrix_cond_exp (f01kgc)

1 Purpose

nag_matop_complex_gen_matrix_cond_exp (f01kgc) computes an estimate of the relative condition number $\kappa_{\exp}(A)$ of the exponential of a complex n by n matrix A, in the 1-norm. The matrix exponential e^A is also returned.

2 Specification

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

```
void nag_matop_complex_gen_matrix_cond_exp (Integer n, Complex a[],
     Integer pda, double *condea, NagError *fail)
```

3 Description

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

$$e^{A+E} - e^A - L(A, E) = o(\|E\|).$$

The derivative describes the first-order effect of perturbations in A on the exponential e^A.

The relative condition number of the matrix exponential can be defined by

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

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

To obtain the estimate of $\kappa_{\exp}(A)$, nag_matop_complex_gen_matrix_cond_exp (f01kgc) first estimates $\|L(A)\|$ by computing an estimate γ of a quantity $K \in \left[n^{-1}\|L(A)\|_1, n\|L(A)\|_1\right]$, such that $\gamma \leq K$.

The algorithms used to compute $\kappa_{\exp}(A)$ are detailed in the Al–Mohy and Higham (2009a) and Al–Mohy and Higham (2009b).

The matrix exponential e^A is computed using a Padé approximant and the scaling and squaring method. The Padé approximant is differentiated to obtain the Fréchet derivatives $L(A, E)$ which are used to estimate the condition number.

4 References

Al–Mohy A H and Higham N J (2009a) A new scaling and squaring algorithm for the matrix exponential *SIAM J. Matrix Anal.* **31(3)** 970–989

Al–Mohy A H and Higham N J (2009b) Computing the Fréchet derivative of the matrix exponential, with an application to condition number estimation *SIAM J. Matrix Anal. Appl.* **30(4)** 1639–1657

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

Moler C B and Van Loan C F (2003) Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later *SIAM Rev.* **45** 3–49

5 Arguments

1: **n** – Integer *Input*

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

Constraint: **n** ≥ 0.

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

Note: the dimension, *dim*, of the array **a** must be at least **pda** $\times$ **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 matrix exponential e^A.

3: **pda** – Integer *Input*

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

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

4: **condea** – double * *Output*

On exit: an estimate of the relative condition number of the matrix exponential $\kappa_{\exp}(A)$.

5: **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 $\langle$*value*$\rangle$ had an illegal value.

NE_INT

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

NE_SINGULAR

The linear equations to be solved for the Padé approximant are singular; it is likely that this function has been called incorrectly.

NW_SOME_PRECISION_LOSS

e^A 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_exp (f01kgc) uses the norm estimation function nag_linsys_complex_gen_norm_rcomm (f04zdc) to produce an estimate γ of a quantity $K \in \left[n^{-1}\|L(A)\|_1, n\|L(A)\|_1\right]$, 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 $A^{\mathrm{H}}A = AA^{\mathrm{H}}$) the computed matrix, e^A, is guaranteed to be close to the exact matrix, that is, the method is forward stable. No such guarantee can be given for non-normal matrices. See Section 10.3 of Higham (2008) for details and further discussion.

For further discussion of the condition of the matrix exponential see Section 10.2 of Higham (2008).

8 Parallelism and Performance

nag_matop_complex_gen_matrix_cond_exp (f01kgc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_matop_complex_gen_matrix_cond_exp (f01kgc) 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

nag_matop_complex_gen_matrix_cond_std (f01kac) uses a similar algorithm to nag_matop_complex_gen_matrix_cond_exp (f01kgc) to compute an estimate of the *absolute* condition number (which is related to the relative condition number by a factor of $\|A\|/\|\exp(A)\|$). However, the required Fréchet derivatives are computed in a more efficient and stable manner by nag_matop_complex_gen_matrix_cond_exp (f01kgc) and so its use is recommended over nag_matop_complex_gen_matrix_cond_std (f01kac).

The cost of the algorithm is $O(n^3)$ and the complex allocatable memory required is approximately $15n^2$; see Al–Mohy and Higham (2009a) and Al–Mohy and Higham (2009b) for further details.

If the matrix exponential alone is required, without an estimate of the condition number, then nag_matop_complex_gen_matrix_exp (f01fcc) should be used. If the Fréchet derivative of the matrix exponential is required then nag_matop_complex_gen_matrix_frcht_exp (f01khc) should be used.

As well as the excellent book Higham (2008), the classic reference for the computation of the matrix exponential is Moler and Van Loan (2003).

10 Example

This example estimates the relative condition number of the matrix exponential e^A, where

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

10.1 Program Text

```
/* nag_matop_complex_gen_matrix_cond_exp (f01kgc) 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]

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

  INIT_FAIL(fail);

  printf("nag_matop_complex_gen_matrix_cond_exp (f01kgc) ");
  printf("Example Program Results\n\n");
  fflush(stdout);

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

  /* Read in the problem size */
  scanf("%ld%*[^\n]", &n);

  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++)
    for (j = 0; j < n; j++)
      scanf(" ( %lf , %lf ) ", &A(i, j).re, &A(i, j).im);
  scanf("%*[^\n]");

  /* Find exp(A) and the condition number using
   * nag_matop_complex_gen_matrix_cond_exp (f01kgc)
   * Condition number for complex matrix exponential
   */
  nag_matop_complex_gen_matrix_cond_exp(n, a, pda, &condea, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_matop_complex_gen_matrix_cond_exp (f01kgc)\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

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

 END:
```

```
  NAG_FREE(a);

  return exit_status;
}
```

10.2 Program Data

```
nag_matop_complex_gen_matrix_cond_exp (f01kgc) Example Program Data

  4                                          :Value of n

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

10.3 Program Results

```
nag_matop_complex_gen_matrix_cond_exp (f01kgc) Example Program Results

 exp(A)
            1           2           3           4
 1   -157.9003   -194.6526   -186.5627   -155.7669
     -754.3717   -555.0507   -475.4533   -520.1876

 2   -206.8899   -225.4985   -212.4414   -186.5627
     -694.7443   -505.3938   -431.0611   -475.4533

 3   -208.7476   -238.4962   -225.4985   -194.6526
     -808.2090   -590.8045   -505.3938   -555.0507

 4   -133.3958   -208.7476   -206.8899   -157.9003
    -1085.5496   -808.2090   -694.7443   -754.3717

Estimated relative condition number is:   15.29
```