NAG Library Function Document

nag_real_symm_matrix_exp (f01edc)

1 Purpose

nag_real_symm_matrix_exp (f01edc) computes the matrix exponential, e^A, of a real symmetric n by n matrix A.

2 Specification

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

```
void nag_real_symm_matrix_exp (Nag_OrderType order, Nag_UploType uplo,
     Integer n, double a[], Integer pda, NagError *fail)
```

3 Description

e^A is computed using a spectral factorization of A

$$A = QDQ^{\mathrm{T}},$$

where D is the diagonal matrix whose diagonal elements, d_i, are the eigenvalues of A, and Q is an orthogonal matrix whose columns are the eigenvectors of A. e^A is then given by

$$e^A = Qe^DQ^{\mathrm{T}},$$

where e^D is the diagonal matrix whose ith diagonal element is e^{d_i}. See for example Section 4.5 of Higham (2008).

4 References

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: **order** – Nag_OrderType *Input*

On entry: the **order** argument specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by **order** = Nag_RowMajor. See Section 2.3.1.3 in How to Use the NAG Library and its Documentation for a more detailed explanation of the use of this argument.

Constraint: **order** = Nag_RowMajor or Nag_ColMajor.

2: **uplo** – Nag_UploType *Input*

On entry: indicates whether the upper or lower triangular part of A is stored.

uplo = Nag_Upper
The upper triangular part of A is stored.

uplo = Nag_Lower
The lower triangular part of A is stored.

Constraint: **uplo** = Nag_Upper or Nag_Lower.

3: **n** – Integer *Input*

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

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

4: **a**$[dim]$ – double *Input/Output*

Note: the dimension, *dim*, of the array **a** must be at least $\mathbf{pda} \times \mathbf{n}$.

On entry: the n by n symmetric matrix A.

If **order** = Nag_ColMajor, A_{ij} is stored in $\mathbf{a}[(j-1) \times \mathbf{pda} + i - 1]$.

If **order** = Nag_RowMajor, A_{ij} is stored in $\mathbf{a}[(i-1) \times \mathbf{pda} + j - 1]$.

If **uplo** = Nag_Upper, the upper triangular part of A must be stored and the elements of the array below the diagonal are not referenced.

If **uplo** = Nag_Lower, the lower triangular part of A must be stored and the elements of the array above the diagonal are not referenced.

On exit: if **fail.code** = NE_NOERROR, the upper or lower triangular part of the n by n matrix exponential, e^A.

5: **pda** – Integer *Input*

On entry: the stride separating row or column elements (depending on the value of **order**) of the matrix A in the array **a**.

Constraint: $\mathbf{pda} \geq \max(1, \mathbf{n})$.

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 ⟨*value*⟩ had an illegal value.

NE_CONVERGENCE

The computation of the spectral factorization failed to converge.

NE_INT

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

On entry, **pda** = ⟨*value*⟩.
Constraint: $\mathbf{pda} > 0$.

NE_INT_2

On entry, **pda** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: $\mathbf{pda} \geq \mathbf{n}$.

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 computing the spectral factorization. Please contact NAG.

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

For a symmetric matrix A, the matrix e^A, has the relative condition number

$$\kappa(A) = \|A\|_2,$$

which is the minimum possible for the matrix exponential and so the computed matrix exponential is guaranteed to be close to the exact matrix. See Section 10.2 of Higham (2008) for details and further discussion.

8 Parallelism and Performance

nag_real_symm_matrix_exp (f01edc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_real_symm_matrix_exp (f01edc) 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 Integer allocatable memory required is **n**, and the double allocatable memory required is approximately $(\mathbf{n} + nb + 4) \times \mathbf{n}$, where nb is the block size required by nag_dsyev (f08fac).

The cost of the algorithm is $O(n^3)$.

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

10 Example

This example finds the matrix exponential of the symmetric matrix

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 2 & 3 \\ 3 & 2 & 1 & 2 \\ 4 & 3 & 2 & 1 \end{pmatrix}$$

10.1 Program Text

```
/* nag_real_symm_matrix_exp (f01edc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

/* Pre-processor includes */
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf01.h>
#include <nagx04.h>

int main(void)
{
  /*Integer scalar and array declarations */
  Integer exit_status = 0;
  Integer i, j, n, pda;
  Nag_MatrixType matrix;
  Nag_UploType uploc;
  /*Double scalar and array declarations */
  double *a = 0;
  /*Character scalar and array declarations */
  char uplo[10];
  Nag_OrderType order;
  NagError fail;

  INIT_FAIL(fail);

  printf("%s\n", "nag_real_symm_matrix_exp (f01edc) Example Program Results");
  printf("\n");
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%*[^\n] ", &n);
#else
  scanf("%" NAG_IFMT "%*[^\n] ", &n);
#endif
#ifdef NAG_COLUMN_MAJOR
  pda = n;
#define A(I, J) a[(J-1)*pda + I-1]
  order = Nag_ColMajor;
#else
  pda = n;
#define A(I, J) a[(I-1)*pda + J-1]
  order = Nag_RowMajor;
#endif
  if (!(a = NAG_ALLOC(n * n, double)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
#ifdef _WIN32
  scanf_s("%9s%*[^\n] ", uplo, (unsigned)_countof(uplo));
#else
  scanf("%9s%*[^\n] ", uplo);
#endif
  /*
   * nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value
```

```
   */
  uploc = (Nag_UploType) nag_enum_name_to_value(uplo);
  if (uploc == Nag_Upper) {
    matrix = Nag_UpperMatrix;
    for (i = 1; i <= n; i++) {
      for (j = i; j <= n; j++)
#ifdef _WIN32
        scanf_s("%lf ", &A(i, j));
#else
        scanf("%lf ", &A(i, j));
#endif
    }
#ifdef _WIN32
    scanf_s("%*[^\n] ");
#else
    scanf("%*[^\n] ");
#endif
  }
  else {
    matrix = Nag_LowerMatrix;
    for (i = 1; i <= n; i++) {
      for (j = 1; j <= i; j++)
#ifdef _WIN32
        scanf_s("%lf ", &A(i, j));
#else
        scanf("%lf ", &A(i, j));
#endif
    }
#ifdef _WIN32
    scanf_s("%*[^\n] ");
#else
    scanf("%*[^\n] ");
#endif
  }
  /*
   * nag_real_symm_matrix_exp (f01edc)
   * Real symmetric matrix exponential
   */
  nag_real_symm_matrix_exp(order, uploc, n, a, pda, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_real_symm_matrix_exp (f01edc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }
  /*
   * nag_gen_real_mat_print (x04cac)
   * Print real general matrix (easy-to-use)
   */
  fflush(stdout);
  nag_gen_real_mat_print(order, matrix, Nag_NonUnitDiag, n, n, a, pda,
                         "Symmetric Exp(A)", 0, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_gen_real_mat_print (x04cac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

END:
  NAG_FREE(a);

  return exit_status;
}
```

10.2 Program Data

```
nag_real_symm_matrix_exp (f01edc) Example Program Data
  4                       :Value of n
  Nag_Upper               :Value of uplo
  1.0   2.0   3.0   4.0
        1.0   2.0   3.0
              1.0   2.0
                    1.0  :End of matrix A
```

10.3 Program Results

```
nag_real_symm_matrix_exp (f01edc) Example Program Results

 Symmetric Exp(A)
            1          2          3          4
 1   2675.3899  2193.0210  2193.2062  2675.2803
 2              1798.3297  1797.8497  2193.2062
 3                         1798.3297  2193.0210
 4                                    2675.3899
```