1 Purpose

nag_real_gen_matrix_exp (f01ecc) computes the matrix exponential, \( e^A \), of a real \( n \times n \) matrix \( A \).

2 Specification

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

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

3 Description

\( e^A \) is computed using a Padé approximant and the scaling and squaring method described in Al–Mohy and Higham (2009).

4 References


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: \( \text{order} \) – Nag_OrderType

   \( \text{Input} \)

   \( \text{On entry:} \) the \( \text{order} \) 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 \( \text{order} = \text{Nag_RowMajor} \). See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

   \( \text{Constraint:} \ \text{order} = \text{Nag_RowMajor} \text{ or Nag_ColMajor}. \)

2: \( n \) – Integer

   \( \text{Input} \)

   \( \text{On entry:} \) \( n \), the order of the matrix \( A \).

   \( \text{Constraint:} \ \ n \geq 0. \)

3: \( a[\text{dim}] \) – double

   \( \text{Input/Output} \)

   \( \text{Note:} \) the dimension, \( \text{dim} \), of the array \( a \) must be at least \( \text{pda} \times n. \)

   The \( (i,j) \)th element of the matrix \( A \) is stored in

   \[ a[(j-1) \times \text{pda} + i - 1] \text{ when } \text{order} = \text{Nag_ColMajor}; \]

   \[ a[(i-1) \times \text{pda} + j - 1] \text{ when } \text{order} = \text{Nag_RowMajor}. \]

   \( \text{On entry:} \) the \( n \) by \( n \) matrix \( A \).
On exit: the \( n \) by \( n \) matrix exponential \( e^A \).

4: \( \text{pda} \) – Integer  

\textit{Input}

\textit{On entry:} the stride separating row or column elements (depending on the value of \textit{order}) in the array \( a \).

\textit{Constraint:} \( \text{pda} \geq n \).

5: \( \text{fail} \) – NagError *  

\textit{Input/Output}

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

\textbf{NE_ALLOC_FAIL}

Dynamic memory allocation failed.

See Section 3.2.1.2 in the Essential Introduction for further information.

\textbf{NE_BAD_PARAM}

On entry, argument \( \langle \text{value} \rangle \) had an illegal value.

\textbf{NE_INT}

On entry, \( n = \langle \text{value} \rangle \).

Constraint: \( n \geq 0 \).

\textbf{NE_INT_2}

On entry, \( \text{pda} = \langle \text{value} \rangle \) and \( n = \langle \text{value} \rangle \).

Constraint: \( \text{pda} \geq n \).

\textbf{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 3.6.6 in the Essential Introduction for further information.

An unexpected internal error has occurred. Please contact NAG.

\textbf{NE_NO_LICENCE}

Your licence key may have expired or may not have been installed correctly.

See Section 3.6.5 in the Essential Introduction for further information.

\textbf{NE_SINGULAR}

The linear equations to be solved are nearly singular and the Padé approximant probably has no correct figures; it is likely that this function has been called incorrectly.

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

\textbf{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

For a normal matrix \( A \) (for which \( A^T A = AA^T \)) 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 Al–Mohy and Higham (2009) and Section 10.3 of Higham (2008) for details and further discussion.

If estimates of the condition number of the matrix exponential are required then nag_matop_real_gen_matrix_cond_exp (f01jgc) should be used.

8 Parallelism and Performance

nag_real_gen_matrix_exp (f01ecc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_real_gen_matrix_exp (f01ecc) 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 \( 6 \times n^2 \).

The cost of the algorithm is \( O(n^3) \); see Section 5 of of Al–Mohy and Higham (2009). The real allocatable memory required is approximately \( 6 \times n^2 \).

If the Fréchet derivative of the matrix exponential is required then nag_matop_real_gen_matrix_frcht_exp (f01jhc) should be used.

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 matrix

\[
A = \begin{pmatrix}
1 & 2 & 2 & 2 \\
3 & 1 & 1 & 2 \\
3 & 2 & 1 & 2 \\
3 & 3 & 3 & 1
\end{pmatrix}
\]

10.1 Program Text

/* nag_real_gen_matrix_exp (f01ecc) Example Program. *
* Copyright 2014 Numerical Algorithms Group.
* Mark 9, 2009.
*/

/* Pre-processor includes */
#include <stdio.h>
#include <math.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;
Integer pda;
NagError fail;
/*Double scalar and array declarations */
double *a = 0;
Nag_OrderType order;

INIT_FAIL(fail);

printf("%sn\n",
"nag_real_gen_matrix_exp (f01ecc) Example Program Results");
printf("\n");

#ifdef _WIN32
scanf_s("%*[^
"]");
#else
scanf("%*[^
"]");
#endif

#ifdef _WIN32
scanf_s("%"NAG_IFMT"%*[^
"]", &n);
#else
scanf("%"NAG_IFMT"%*[^
"]", &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(pda*n, double)))
{
printf("Allocation failure\n");
exit_status = -1;
go to END;
}

/* Read A from data file*/
for (i = 1; i <= n; i++)
{
    for (j = 1; j <= n; j++)
        #ifdef _WIN32
scanf_s("%lf", &A(i, j));
#else
scanf("%lf", &A(i, j));
#endif
        #ifdef _WIN32
scanf_s("%*[^
"]");
#else
scanf("%*[^
"]");
#endif
    /* Find exp( A )*/
    /*
    * nag_real_gen_matrix_exp (f01ecc)
    * Real matrix exponential
    */
    nag_real_gen_matrix_exp(order, n, a, pda, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_real_gen_matrix_exp (f01ecc).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }
    /* Print solution*/
    /*
    * nag_gen_real_mat_print (x04cac)
    */
```c
* Print real general matrix (easy-to-use)
*/
fflush(stdout);
nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, n,
a, pda, "Exp(A)", 0, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_real_mat_print (x04cac).\n\n", fail.message);
    exit_status = 1;
}
END:
NAG_FREE(a);
return exit_status;
}

10.2 Program Data

nag_real_gen_matrix_exp (f01ecc) Example Program Data

4 :Value of n

1.0 2.0 2.0 2.0
3.0 1.0 1.0 2.0
3.0 2.0 1.0 2.0
3.0 3.0 3.0 1.0 :End of matrix A

10.3 Program Results

nag_real_gen_matrix_exp (f01ecc) Example Program Results

Exp(A)

1 740.7038 610.8500 542.2743 549.1753
2 731.2510 603.5524 535.0884 542.2743
3 823.7630 679.4257 603.5524 610.8500
4 998.4355 823.7630 731.2510 740.7038
```