NAG Library Function Document

nag_dtrtri (f07tjc)

1 Purpose
nag_dtrtri (f07tjc) computes the inverse of a real triangular matrix.

2 Specification
#include <nag.h>
#include <nagf07.h>
void nag_dtrtri (Nag_OrderType order, Nag_UploType uplo, Nag_DiagType diag,
               Integer n, double a[], Integer pda, NagError *fail)

3 Description
nag_dtrtri (f07tjc) forms the inverse of a real triangular matrix $A$. Note that the inverse of an upper (lower) triangular matrix is also upper (lower) triangular.

4 References

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 3.2.1.3 in the Essential Introduction 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: specifies whether $A$ is upper or lower triangular.
   $uplo = Nag_Upper$
   $A$ is upper triangular.
   $uplo = Nag_Lower$
   $A$ is lower triangular.
   Constraint: uplo = Nag_Upper or Nag_Lower.

3: diag – Nag_DiagType
   Input
   On entry: indicates whether $A$ is a nonunit or unit triangular matrix.
   $diag = Nag_NonUnitDiag$
   $A$ is a nonunit triangular matrix.
   $diag = Nag_UnitDiag$
   $A$ is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.
   Constraint: diag = Nag_NonUnitDiag or Nag_UnitDiag.
4:  

Integ  

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

Constraint: \( n \geq 0 \).

5:  

\( a[dim] \) – double

Input/Output

Note: the dimension, \( dim \), of the array \( a \) must be at least \( \max(1, pda \times n) \).

On entry: the \( n \) by \( n \) triangular matrix \( A \).

If \( order = \text{Nag}\_\text{ColMajor} \), \( A_{ij} \) is stored in \( a[(j - 1) \times pda + i - 1] \).

If \( order = \text{Nag}\_\text{RowMajor} \), \( A_{ij} \) is stored in \( a[(i - 1) \times pda + j - 1] \).

If \( uplo = \text{Nag}\_\text{Upper} \), the upper triangular part of \( A \) must be stored and the elements of the array below the diagonal are not referenced.

If \( uplo = \text{Nag}\_\text{Lower} \), the lower triangular part of \( A \) must be stored and the elements of the array above the diagonal are not referenced.

If \( diag = \text{Nag}\_\text{UnitDiag} \), the diagonal elements of \( A \) are assumed to be \( 1 \), and are not referenced.

On exit: \( A \) is overwritten by \( A^{-1} \), using the same storage format as described above.

6:  

\( pda \) – 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: \( pda \geq \max(1, n) \).

7:  

\( fail \) – 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.

See Section 3.2.1.2 in the Essential Introduction 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: \( n \geq 0 \).

On entry, \( pda = \langle value \rangle \).

Constraint: \( pda > 0 \).

NE_INT_2

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

Constraint: \( pda \geq \max(1, n) \).

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 3.6.6 in the Essential Introduction for further information.

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.

NE_SINGULAR
Element (value) of the diagonal is exactly zero. A is singular its inverse cannot be computed.

7 Accuracy
The computed inverse $X$ satisfies

$$|XA - I| \leq c(n)\epsilon |X||A|,$$

where $c(n)$ is a modest linear function of $n$, and $\epsilon$ is the machine precision.

Note that a similar bound for $|AX - I|$ cannot be guaranteed, although it is almost always satisfied. The computed inverse satisfies the forward error bound

$$|X - A^{-1}| \leq c(n)\epsilon |A^{-1}||A||X|.$$


8 Parallelism and Performance
nag_dtrtri (f07tjc) is not threaded by NAG in any implementation.

nag_dtrtri (f07tjc) 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 total number of floating-point operations is approximately $\frac{1}{3}n^3$.

The complex analogue of this function is nag_zttrtri (f07twc).

10 Example
This example computes the inverse of the matrix $A$, where

$$A = \begin{pmatrix}
4.30 & 0.00 & 0.00 & 0.00 \\
-3.96 & -4.87 & 0.00 & 0.00 \\
0.40 & 0.31 & -8.02 & 0.00 \\
-0.27 & 0.07 & -5.95 & 0.12
\end{pmatrix}.$$

10.1 Program Text
/* nag_dtrtri (f07tjc) Example Program. */
/* Copyright 2014 Numerical Algorithms Group. */
/* * Mark 7, 2001. */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    Integer i, j, n, pda;
    Integer exit_status = 0;
    Nag_UploType uplo;
    Nag_MatrixType matrix;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    char nag_enum_arg[40];
    double *a = 0;
    #ifdef NAG_LOAD_FP
    /* The following line is needed to force the Microsoft linker
to load floating point support */
    float force_loading_of_ms_float_support = 0;
    #endif /* NAG_LOAD_FP */
    #ifdef NAG_COLUMN_MAJOR
    #define A(I, J) a[(J-1)*pda +I-1 ]
    order = Nag_ColMajor;
    #else
    #define A(I, J) a[(I-1)*pda +J-1 ]
    order = Nag_RowMajor;
    #endif
    INIT_FAIL(fail);
    printf("nag_dtrtri (f07tjc) Example Program Results\n\n");
    /* Skip heading in data file */
    #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;
    #else
    pda = n;
    #endif
    /* Allocate memory */
    if (!(a = NAG_ALLOC(n * n, double)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    /* Read A from data file */
    #ifdef _WIN32
    scanf_s(" %39s%*[\n] ", nag_enum_arg, _countof(nag_enum_arg));
    #else
    scanf(" %39s%*[\n] ", nag_enum_arg);
    #endif
    /* nag_enum_name_to_value (x04nac).
    * Converts NAG enum member name to value */
    uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);
    if (uplo == 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
}
/* Compute inverse of A */
/* nag_dtrtri (f07tjc).  *
 * Inverse of real triangular matrix  *
*/
nag_dtrtri(order, uplo, Nag_NonUnitDiag, n, a, pda, &fail);
if (fail.code != NE_NOERROR)
  {
    printf("Error from nag_dtrtri (f07tjc).\n\n", fail.message);
    exit_status = 1;
    goto END;
  }
/* Print inverse */
/* 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,
                      "Inverse", 0, &fail);
if (fail.code != NE_NOERROR)
  {
    printf("Error from nag_gen_real_mat_print (x04cac).\n\n", fail.message);
    exit_status = 1;
    goto END;
  }
END:
NAG_FREE(a);
return exit_status;
10.2 Program Data

nag_dtrtri (f07tjc) Example Program Data

- Value of n
4
- Value of uplo
Nag_Lower

4.30
-3.96 -4.87
0.40 0.31 -8.02
-0.27 0.07 -5.95 0.12

: End of matrix A

10.3 Program Results

nag_dtrtri (f07tjc) Example Program Results

Inverse

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.2326</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-0.1891</td>
<td>-0.2053</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.0043</td>
<td>-0.0079</td>
<td>-0.1247</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.8463</td>
<td>-0.2738</td>
<td>-6.1825</td>
<td>8.3333</td>
</tr>
</tbody>
</table>