NAG C Library Function Document

nag_dtrtri (f07tjc)

1 Purpose

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

2 Specification

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 Parameters

1: order – Nag_OrderType

   On entry: the order parameter 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 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

   Constraint: order = Nag_RowMajor or Nag_ColMajor.

2: uplo – Nag_UploType

   On entry: indicates whether A is upper or lower triangular as follows:
     if uplo = Nag_Upper, A is upper triangular;
     if uplo = Nag_Lower, A is lower triangular.

   Constraint: uplo = Nag_Upper or Nag_Lower.

3: diag – Nag_DiagType

   On entry: indicates whether A is a non-unit or unit triangular matrix as follows:
     if diag = Nag_NonUnitDiag, A is a non-unit triangular matrix;
     if 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: n – Integer

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

   Constraint: n ≥ 0.
5: \(a[\text{dim}]\) – double

**Input/Output**

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

If \(\text{order} = \text{Nag\_ColMajor}\), the \((i, j)\)th element of the matrix \(A\) is stored in \(a[(j - 1) \times \text{pda} + i - 1]\) and if \(\text{order} = \text{Nag\_RowMajor}\), the \((i, j)\)th element of the matrix \(A\) is stored in \(a[(i - 1) \times \text{pda} + j - 1]\).

**On entry:** the \(n\) by \(n\) triangular matrix \(A\). If \(\text{uplo} = \text{Nag\_Upper}\), \(A\) is upper triangular and the elements of the array below the diagonal are not referenced; if \(\text{uplo} = \text{Nag\_Lower}\), \(A\) is lower triangular and the elements of the array above the diagonal are not referenced. If \(\text{diag} = \text{Nag\_UnitDiag}\), the diagonal elements of \(A\) are not referenced, but are assumed to be 1.

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

6: \(\text{pda}\) – Integer

**Input**

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

**Constraint:** \(\text{pda} \geq \max(1, n)\).

7: \(\text{fail}\) – NagError *

**Output**

The NAG error parameter (see the Essential Introduction).

6  **Error Indicators and Warnings**

**NE\_INT**

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

Constraint: \(n \geq 0\).

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

Constraint: \(\text{pda} > 0\).

**NE\_INT\_2**

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

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

**NE\_SINGULAR**

\(a(\langle\text{value}\rangle, \langle\text{value}\rangle)\) is zero, and the matrix \(A\) is singular.

**NE\_ALLOC\_FAIL**

Memory allocation failed.

**NE\_BAD\_PARAM**

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

**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 consult NAG for assistance.

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

\[ |X - A^{-1}| \leq c(n)\|A^{-1}\|\|A\|\|X\|. \]


8 Further Comments

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

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

9 Example

To compute the inverse of the matrix \(A\), where

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

9.1 Program Text

/* nag_dtrtri (f07tjc) Example Program. */
* Copyright 2001 Numerical Algorithms Group.
*/
#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_enum;
    Nag_MatrixType matrix;

    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    char uplo[2];
    double *a=0;

    #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);
    Vprintf("f07tjc Example Program Results\n\n");
    /* Skip heading in data file */
    Vscanf("%*
\n");
    Vscanf("%ld%*
\n", &n);
    #ifdef NAG_COLUMN_MAJOR
    pda = n;
    #else
    pda = n;

    [NP3645/7]
#endif
/* Allocate memory */
if (!(a = NAG_ALLOC(n * n, double)))
  {
    Vprintf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
/* Read A from data file */
Vscanf("' %ls ' %*[\n] ", uplo);
if (*((unsigned char *)uplo == 'L')
  {
    uplo_enum = Nag_Lower;
    matrix = Nag_LowerMatrix;
  }
else if (*((unsigned char *)uplo == 'U')
  {
    uplo_enum = Nag_Upper;
    matrix = Nag_UpperMatrix;
  }
else
  {
    Vprintf("Unrecognised character for Nag_UploType type\n");
    exit_status = -1;
    goto END;
  }
if (uplo_enum == Nag_Upper)
  {
    for (i = 1; i <= n; ++i)
      {
        for (j = i; j <= n; ++j)
          Vscanf("%lf", &A(i,j));
    };
    Vscanf("%*[\n] ");
  }
else
  {
    for (i = 1; i <= n; ++i)
      {
        for (j = 1; j <= i; ++j)
          Vscanf("%lf", &A(i,j));
    };
    Vscanf("%*[\n] ");
  }
/* Compute inverse of A */
f07tjc(order, uplo_enum, Nag_NonUnitDiag, n, a, pda, &fail);
if (fail.code != NE_NOERROR)
  {
    Vprintf("Error from f07tjc.\n\n", fail.message);
    exit_status = 1;
    goto END;
  }
/* Print inverse */
x04cac(order, matrix, Nag_NonUnitDiag, n, n, a, pda,
  "Inverse", 0, &fail);
if (fail.code != NE_NOERROR)
  {
    Vprintf("Error from x04cac.\n\n", fail.message);
    exit_status = 1;
    goto END;
  }
END:
if (a) NAG_FREE(a);
return exit_status;
}
9.2 Program Data

f07tjc Example Program Data
4
'L'
4.30
-3.96 -4.87
0.40 0.31 -8.02
-0.27 0.07 -5.95 0.12 : End of matrix A

9.3 Program Results

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>

f07 – Linear Equations (LAPACK)