NAG C Library Function Document

nag_dtptri (f07ujc)

1 Purpose

nag_dtptri (f07ujc) computes the inverse of a real triangular matrix, using packed storage.

2 Specification

void nag_dtptri (Nag_OrderType order, Nag_UploType uplo, Nag_DiagType diag,
                   Integer n, double ap[], NagError *fail)

3 Description

nag_dtptri (f07ujc) forms the inverse of a real triangular matrix A using packed storage. Note that the inverse of an upper (lower) triangular matrix is also upper (lower) triangular.

4 References


5 Parameters

1: order – Nag_OrderType
   Input
   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
   Input
   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
   Input
   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
   Input
   On entry: n, the order of the matrix A.
   Constraint: n ≥ 0.
5: \( \text{ap}[\dim] \) – double

\textbf{Input/Output}

\textit{Note:} the dimension, \( \dim \), of the array \( \text{ap} \) must be at least \( \max(1, n \times (n + 1)/2) \).

\textit{On entry:} the \( n \) by \( n \) triangular matrix \( A \), packed by rows or columns. The storage of elements \( a_{ij} \) depends on the \texttt{order} and \texttt{uplo} parameters as follows:

- \textbf{if} \( \texttt{order} = \text{Nag\_ColMajor} \) and \( \texttt{uplo} = \text{Nag\_Upper} \),
  \( a_{ij} \) is stored in \( \text{ap}[(j-1) \times j/2 + i] \), for \( i \leq j \);
- \textbf{if} \( \texttt{order} = \text{Nag\_ColMajor} \) and \( \texttt{uplo} = \text{Nag\_Lower} \),
  \( a_{ij} \) is stored in \( \text{ap}[(2n-j) \times (j-1)/2 + i] \), for \( i \geq j \);
- \textbf{if} \( \texttt{order} = \text{Nag\_RowMajor} \) and \( \texttt{uplo} = \text{Nag\_Upper} \),
  \( a_{ij} \) is stored in \( \text{ap}[(2n-i) \times (i-1)/2 + j] \), for \( i \leq j \);
- \textbf{if} \( \texttt{order} = \text{Nag\_RowMajor} \) and \( \texttt{uplo} = \text{Nag\_Lower} \),
  \( a_{ij} \) is stored in \( \text{ap}[(i-1) \times i/2 + j] \), for \( i \geq j \).

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

6: \( \text{fail} \) – NagError *

\textbf{Output}

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

\textbf{NE\_INT}

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

Constraint: \( n \geq 0 \).

\textbf{NE\_SINGULAR}

The matrix \( A \) is singular.

\textbf{NE\_ALLOC\_FAIL}

Memory allocation failed.

\textbf{NE\_BAD\_PARAM}

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

\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 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 \textit{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 Further Comments

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

The complex analogue of this function is nag_ztptri (f07uwc).

9 Example

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

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

using packed storage.

9.1 Program Text

/* nag_dtptri (f07ujc) Example Program. */
/* Copyright 2001 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 ap_len, i, j, n;
    Integer exit_status=0;
    NagError fail;
    Nag_UploType uplo_enum;
    Nag_OrderType order;
    /* Arrays */
    char uplo[2];
    double *ap=0;
    #ifdef NAG_COLUMN_MAJOR
    #define A_UPPER(I,J) ap[J*(J-1)/2 + I - 1]
    #define A_LOWER(I,J) ap[(2*n-J)*(J-1)/2 + I - 1]
    order = Nag_ColMajor;
    #else
    #define A_LOWER(I,J) ap[I*(I-1)/2 + J - 1]
    #define A_UPPER(I,J) ap[(2*n-I)*(I-1)/2 + J - 1]
    order = Nag_RowMajor;
    #endif

    INIT_FAIL(fail);
    Vprintf("f07ujc Example Program Results\n\n");
    /* Skip heading in data file */
    Vscanf("%*[^
\] ");
    Vscanf("%ld%*[^
\] ", &n);
    ap_len = n * (n + 1)/2;

    /* Allocate memory */
    if ( !(ap = NAG_ALLOC(ap_len, 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;
else if (*(unsigned char *)uplo == 'U')
    uplo_enum = Nag_Upper;
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_UPPER(i,j));
    }
    Vscanf("%*[\n] ");
}
else
{
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= i; ++j)
            Vscanf("%lf", &A_LOWER(i,j));
    }
    Vscanf("%*[\n] ");
}

/* Compute inverse of A */
f07ujc(order, uplo_enum, Nag_NonUnitDiag, n, ap, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07ujc.\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print inverse */
x04ccc(order, uplo_enum, Nag_NonUnitDiag, n, ap,
    "Inverse", 0, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from x04ccc.\n", fail.message);
    exit_status = 1;
    goto END;
}

END:
if (ap) NAG_FREE(ap);
return exit_status;

9.2 Program Data

f07ujc Example Program Data

4 : Value of N
'L' : Value of UPLO
-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

f07ujc Example Program Results

<table>
<thead>
<tr>
<th>Inverse</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>