NAG C Library Function Document

nag_ztptri (f07uwc)

1 Purpose

nag_ztptri (f07uwc) computes the inverse of a complex triangular matrix, using packed storage.

2 Specification

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

3 Description

nag_ztptri (f07uwc) forms the inverse of a complex 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: \( \text{ap}[\text{dim}] \) – Complex

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

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

- If \text{order} = Nag_ColMajor and \text{uplo} = Nag_Upper, \( a_{ij} \) is stored in \( \text{ap}[(j - 1) \times j/2 + i - 1] \), for \( i \leq j \);
- If \text{order} = Nag_ColMajor and \text{uplo} = Nag_Lower, \( a_{ij} \) is stored in \( \text{ap}[(2n - j) \times (j - 1)/2 + i - 1] \), for \( i \geq j \);
- If \text{order} = Nag_RowMajor and \text{uplo} = Nag_Upper, \( a_{ij} \) is stored in \( \text{ap}[(i - 1) \times i/2 + j - 1] \), for \( i \leq j \);
- If \text{order} = Nag_RowMajor and \text{uplo} = Nag_Lower, \( a_{ij} \) is stored in \( \text{ap}[(i - 1) \times i/2 + j - 1] \), for \( i \geq j \).

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

6: \text{fail} – NagError *

The NAG error parameter (see the Essential Introduction).

### 6 Error Indicators and Warnings

**NE_INT**

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

Constraint: \( n \geq 0 \).

**NE_SINGULAR**

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

\[
|XA - I| \leq c(n)\varepsilon|X||A|,
\]

where \( c(n) \) is a modest linear function of \( n \), and \( \varepsilon \) 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

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

8 Further Comments

The total number of real floating-point operations is approximately $\frac{4}{3}n^3$.
The real analogue of this function is nag_dptri (f07ujc).

9 Example

To compute the inverse of the matrix $A$, where

$$A = \begin{pmatrix}
4.78 + 4.56i & 0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i \\
2.00 - 0.30i & -4.11 + 1.25i & 0.00 + 0.00i & 0.00 + 0.00i \\
2.89 - 1.34i & 2.36 - 4.25i & 4.15 + 0.80i & 0.00 + 0.00i \\
-1.89 + 1.15i & 0.04 - 3.69i & -0.02 + 0.46i & 0.33 - 0.26i \\
\end{pmatrix},$$

using packed storage.

9.1 Program Text

/* nag_ztptri (f07uwc) 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 ap_len, i, j, n;
    Integer exit_status=0;
    Nag_UploType uplo_enum;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    char uplo[2];
    Complex *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("f07uwc 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, Complex)) )
    {
        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 , %lf )", &A_UPPER(i,j).re, &A_UPPER(i,j).im);
  }
  Vscanf("%*[\n] ");
}
else
{
  for (i = 1; i <= n; ++i)
  {
    for (j = 1; j <= i; ++j)
      Vscanf(" ( %lf , %lf )", &A_LOWER(i,j).re, &A_LOWER(i,j).im);
  }
  Vscanf("%*[\n] ");
}

/* Compute inverse of A */
f07uwc(order, uplo_enum, Nag_NonUnitDiag, n, ap, &fail);
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from f07uwc.\n%s\n", fail.message);
  exit_status = 1;
  goto END;
}

/* Print inverse */
x04ddc(order, uplo_enum, Nag_NonUnitDiag, n, ap,
    Nag_BracketForm, "%7.4f", "Inverse", Nag_IntegerLabels,
    0, Nag_IntegerLabels, 0, 80, 0, 0, &fail);
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from x04ddc.\n%s\n", fail.message);
  exit_status = 1;
  goto END;
}
END:
if (ap) NAG_FREE(ap);
return exit_status;

9.2 Program Data

f07uwc Example Program Data

4 :Value of N
'L' :Value of UPLO
( 4.78, 4.56) ( 2.00,-0.30) (-4.11, 1.25) ( 2.89,-1.34) ( 2.36,-4.25) ( 4.15, 0.80) (-1.89, 1.15) ( 0.04,-3.69) (-0.02, 0.46) ( 0.33,-0.26) :End of matrix A
## 9.3 Program Results

f07uwc 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>(0.1095, -0.1045)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(0.0582, -0.0411)</td>
<td>(-0.2227, -0.0677)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(0.0032, 0.1905)</td>
<td>(0.1538, -0.2192)</td>
<td>(0.2323, -0.0448)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>(0.7602, 0.2814)</td>
<td>(1.6184, -1.4346)</td>
<td>(0.1289, -0.2250)</td>
<td>(1.8697, 1.4731)</td>
</tr>
</tbody>
</table>