NAG Library Function Document

nag_ztftri (f07wxc)

1 Purpose

nag_ztftri (f07wxc) computes the inverse of a complex triangular matrix stored in Rectangular Full Packed (RFP) format.

2 Specification

```c
#include <nag.h>
#include <nagf07.h>

void nag_ztftri (Nag_OrderType order, Nag_RFP_Store transr,
                Nag_UploType uplo, Nag_DiagType diag, Integer n, Complex ar[],
                NagError *fail)
```

3 Description

nag_ztftri (f07wxc) forms the inverse of a complex triangular matrix $A$, stored using RFP format. The RFP storage format is described in Section 3.3.3 in the f07 Chapter Introduction. Note that the inverse of an upper (lower) triangular matrix is also upper (lower) triangular.

4 References


5 Arguments

1: order – Nag_OrderType

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: transr – Nag_RFP_Store

On entry: specifies whether the normal RFP representation of $A$ or its conjugate transpose is stored.

transr = Nag_RFP_Normal
The matrix $A$ is stored in normal RFP format.

transr = Nag_RFP_ConjTrans
The conjugate transpose of the RFP representation of the matrix $A$ is stored.

Constraint: transr = Nag_RFP_Normal or Nag_RFP_ConjTrans.
3: **uplo** – Nag_UplType

*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.*

4: **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.*

5: **n** – Integer

*Input*

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

*Constraint: n ≥ 0.*

6: **ar[n × (n + 1)/2]** – Complex

*Input/Output*

*On entry:* the upper or lower triangular part (as specified by uplo) of the $n$ by $n$ Hermitian matrix $A$, in either normal or transposed RFP format (as specified by transr). The storage format is described in detail in Section 3.3.3 in the f07 Chapter Introduction.

*On exit:* $A$ is overwritten by $A^{-1}$, in the same storage format as $A$.

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 *value* had an illegal value.

**NE_INT**

On entry, $n = *value*.$

*Constraint: n ≥ 0.*

**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.
Diagonal element (value) of $A$ 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_ztftri (f07wxc) is not threaded by NAG in any implementation.

nag_ztftri (f07wxc) 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 real floating-point operations is approximately $\frac{2}{3}n^3$.

The real analogue of this function is nag_dtftri (f07wkc).

10 Example

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

$$A = \begin{pmatrix}
4.78 + 4.56i & 0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i \\
2.09 - 0.32i & -4.11 + 1.51i & 0.00 + 0.00i & 0.00 + 0.00i \\
2.89 - 1.34i & 2.36 - 2.54i & 4.15 + 0.80i & 0.00 + 0.00i \\
-1.89 + 1.15i & 0.04 - 3.69i & -0.02 + 0.46i & 0.33 - 0.26i
\end{pmatrix}$$

and is stored using RFP format.

10.1 Program Text

/* nag_ztftri (f07wxc) Example Program. */
/* Copyright 2014 Numerical Algorithms Group. */
/* Mark 25, 2014. */
*/
#include <nag.h>
#include <nag_stdlib.h>
int main(void)
{
    /* Scalars*/
    Integer exit_status = 0;
    Integer i, j, k, lar1, lar2, lenar, n, pdar, pda, q;
    /* Arrays*/
    Complex *ar = 0, *a = 0;
    char nag_enum_arg[40];
    /* NAG types */
    Nag_RFP_Store transr;
    Nag_UploType uplo;
    Nag_DiagType diag;
    Nag_OrderType order;
    Nag_MatrixType matrix;
    NagError fail;
    if (NAG_COLUMN_MAJOR)
        order = Nag_ColMajor;
    #define AR(I,J) ar[J*pdar + I]
    else
        order = Nag_RowMajor;
    #define AR(I,J) ar[I*pdar + J]
    INIT_FAIL(fail);
    printf("nag_ztftri (f07wxc) 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);
    #else
        scanf("%"NAG_IFMT"", &n);
    #endif
    #ifdef _WIN32
        scanf_s("%39s", nag_enum_arg, _countof(nag_enum_arg));
    #else
        scanf("%39s", nag_enum_arg);
    #endif
    uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);
    #ifdef _WIN32
        scanf_s("%39s", nag_enum_arg, _countof(nag_enum_arg));
    #else
        scanf("%39s", nag_enum_arg);
    #endif
    transr = (Nag_RFP_Store) nag_enum_name_to_value(nag_enum_arg);
    #ifdef _WIN32
        scanf_s("%39s%*[\n] ", nag_enum_arg, _countof(nag_enum_arg));
    #else
        scanf("%39s%*[\n] ", nag_enum_arg);
    #endif
    diag = (Nag_DiagType) nag_enum_name_to_value(nag_enum_arg);
    lenar = (n * (n + 1))/2;
    pda = n;
    if (!((ar = NAG_ALLOC(lenar, Complex)) ||
        !(a = NAG_ALLOC((pda)*(n), Complex))
    )
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    ...
/* Setup dimensions for RFP array ar. */
k = n/2;
q = n - k;
if (transr==Nag_RFP_Normal) {
lar1 = 2*k+1;
lar2 = q;
} else {
lar1 = q;
lar2 = 2*k+1;
}
if (order==Nag_RowMajor) {
pdar = lar2;
} else {
pdar = lar1;
}
/* Read matrix into RFP array ar. */
for (i = 0; i < lar1; i++) {
for (j = 0; j < lar2; j++) {
#ifdef _WIN32
scanf_s(" ( %lf , %lf ) ", &AR(i,j).re, &AR(i,j).im);
#else
scanf(" ( %lf , %lf ) ", &AR(i,j).re, &AR(i,j).im);
#endif
}
/* Compute inverse of A using nag_ztftri (f07wxc). */
nag_ztftri(order, transr, uplo, diag, n, ar, &fail);
if (fail.code != NE_NOERROR) {
printf("%s
", fail.message);
exit_status = 1;
go to END;
}
/* Convert inverse to full matrix format using nag_ztfttr (f01vhc). */
nag_ztfttr(order, transr, uplo, n, ar, a, pda, &fail);
if (fail.code != NE_NOERROR) {
printf("%s
", fail.message);
exit_status = 2;
go to END;
}
/* nag_gen_complex_mat_print_comp (x04dbc).
 * Print complex general matrix (comprehensive)
 */
if (uplo == Nag_Lower)
  matrix = Nag_LowerMatrix;
else
  matrix = Nag_UpperMatrix;
nag_gen_complex_mat_print_comp(order, matrix, diag, n, n, a, pda,
  NagBracketForm, "%7.4f", "Inverse",
  Nag_IntegerLabels, NULL, Nag_IntegerLabels,
  NULL, 80, 0, 0, &fail);
if (fail.code != NE_NOERROR) {
  printf("Error from nag_gen_complex_mat_print_comp (x04dbc)\n%s\n", fail.message);
  exit_status = 3;
}
END:
NAG_FREE(ar);
NAG_FREE(a);
return exit_status;
10.2 Program Data

nag_ztftri (f07wxc) Example Program Data

4
Nag_Lower
Nag_RFP_Normal
Nag_NonUnitDiag

: n, uplo, transr, diag

(  4.15, -0.80)  (-0.02, -0.46)
(  4.78,  4.56)  ( 0.33,  0.26)
(  2.00, -0.30)  (-4.11,  1.25)
(  2.89, -1.34)  (  2.36, -4.25)
(-1.89,  1.15)  (  0.04, -3.69) : ar[]

10.3 Program Results

nag_ztftri (f07wxc) 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>( 0.1095, -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>