

NAG Library Function Document

nag_ztptri (f07uwc)

1 Purpose

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

2 Specification

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

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

Du Croz J J and Higham N J (1992) Stability of methods for matrix inversion *IMA J. Numer. Anal.* **12** 1–19

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: **n** – Integer *Input*
On entry: n , the order of the matrix A .
Constraint: $n \geq 0$.
- 5: **ap**[*dim*] – Complex *Input/Output*
Note: the dimension, *dim*, of the array **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 **order** and **uplo** arguments as follows:
- if **order** = Nag_ColMajor and **uplo** = Nag_Upper,
 A_{ij} is stored in **ap**[($j - 1$) \times $j/2 + i - 1$], for $i \leq j$;
 - if **order** = Nag_ColMajor and **uplo** = Nag_Lower,
 A_{ij} is stored in **ap**[($2n - j$) \times ($j - 1$)/2 + $i - 1$], for $i \geq j$;
 - if **order** = Nag_RowMajor and **uplo** = Nag_Upper,
 A_{ij} is stored in **ap**[($2n - i$) \times ($i - 1$)/2 + $j - 1$], for $i \leq j$;
 - if **order** = Nag_RowMajor and **uplo** = Nag_Lower,
 A_{ij} is stored in **ap**[($i - 1$) \times $i/2 + j - 1$], for $i \geq j$.
- If **diag** = Nag_UnitDiag, the diagonal elements of AP are assumed to be 1, and are not referenced; the same storage scheme is used whether **diag** = Nag_NonUnitDiag or **diag** = Nag_UnitDiag.
On exit: A is overwritten by A^{-1} , using the same storage format as described above.
- 6: **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: $n \geq 0$.

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 $\langle value \rangle$ 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 ϵ 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

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

See Du Croz and Higham (1992).

8 Parallelism and Performance

nag_ztptri (f07uwc) is not threaded by NAG in any implementation.

nag_ztptri (f07uwc) 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 $\frac{4}{3}n^3$.

The real analogue of this function is nag_dtptri (f07ujc).

10 Example

This example computes the inverse of the matrix A , where

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

using packed storage.

10.1 Program Text

```

/* nag_ztptri (f07uwc) 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    ap_len, i, j, n;
    Integer    exit_status = 0;
    Nag_UploType uplo;

```

```

NagError      fail;
Nag_OrderType order;
/* Arrays */
char          nag_enum_arg[40];
Complex       *ap = 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_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);

printf("nag_ztptri (f07uwc) 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
ap_len = n * (n + 1)/2;

/* Allocate memory */
if (!(ap = NAG_ALLOC(ap_len, Complex)))
{
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)
{
for (i = 1; i <= n; ++i)
{
for (j = i; j <= n; ++j)
#ifdef _WIN32
scanf_s(" ( %lf , %lf )", &A_UPPER(i, j).re,
&A_UPPER(i, j).im);
#else
scanf(" ( %lf , %lf )", &A_UPPER(i, j).re,
&A_UPPER(i, j).im);
#endif
}
}

```

```

#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif
}
else
{
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= i; ++j)
#ifdef _WIN32
            scanf_s(" ( %lf , %lf )", &A_LOWER(i, j).re,
                    &A_LOWER(i, j).im);
#else
            scanf(" ( %lf , %lf )", &A_LOWER(i, j).re,
                    &A_LOWER(i, j).im);
#endif
    }
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif
}

/* Compute inverse of A */
/* nag_ztptri (f07uwc).
 * Inverse of complex triangular matrix, packed storage
 */
nag_ztptri(order, uplo, Nag_NonUnitDiag, n, ap, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_ztptri (f07uwc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print inverse */
/* nag_pack_complx_mat_print_comp (x04ddc).
 * Print complex packed triangular matrix (comprehensive)
 */
fflush(stdout);
nag_pack_complx_mat_print_comp(order, uplo, Nag_NonUnitDiag, n, ap,
                               Nag_BracketForm, "%7.4f", "Inverse",
                               Nag_IntegerLabels, 0, Nag_IntegerLabels, 0,
                               80, 0, 0, &fail);

if (fail.code != NE_NOERROR)
{
    printf("Error from nag_pack_complx_mat_print_comp (x04ddc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
}
END:
NAG_FREE(ap);

return exit_status;
}

```

10.2 Program Data

```

nag_ztptri (f07uwc) Example Program Data
4                                     :Value of n
Nag_Lower                            :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

```

10.3 Program Results

nag_ztptri (f07uwc) Example Program Results

```
Inverse
          1              2              3              4
1 ( 0.1095,-0.1045)
2 ( 0.0582,-0.0411) (-0.2227,-0.0677)
3 ( 0.0032, 0.1905) ( 0.1538,-0.2192) ( 0.2323,-0.0448)
4 ( 0.7602, 0.2814) ( 1.6184,-1.4346) ( 0.1289,-0.2250) ( 1.8697, 1.4731)
```
