

NAG C Library Function Document

nag_ztrtrs (f07tsc)

1 Purpose

nag_ztrtrs (f07tsc) solves a complex triangular system of linear equations with multiple right-hand sides, $AX = B$, $A^T X = B$ or $A^H X = B$.

2 Specification

```
void nag_ztrtrs (Nag_OrderType order, Nag_UploType uplo, Nag_TransType trans,
                 Nag_DiagType diag, Integer n, Integer nrhs, const Complex a[], Integer pda,
                 Complex b[], Integer pdb, NagError *fail)
```

3 Description

nag_ztrtrs (f07tsc) solves a complex triangular system of linear equations $AX = B$, $A^T X = B$ or $A^H X = B$.

4 References

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

Higham N J (1989) The accuracy of solutions to triangular systems *SIAM J. Numer. Anal.* **26** 1252–1265

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: **trans** – Nag_TransType *Input*
On entry: indicates the form of the equations as follows:
 if **trans** = **Nag_NoTrans**, then the equations are of the form $AX = B$;
 if **trans** = **Nag_Trans**, then the equations are of the form $A^T X = B$;
 if **trans** = **Nag_ConjTrans**, then the equations are of the form $A^H X = B$.
Constraint: **trans** = **Nag_NoTrans**, **Nag_Trans** or **Nag_ConjTrans**.

- 4: **diag** – Nag_DiagType *Input*
On entry: indicates whether A is a non-unit or unit triangular matrix as follows:
 if **diag** = **Nag_NonUnitDiag**, then A is a non-unit triangular matrix;
 if **diag** = **Nag_UnitDiag**, then 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: $n \geq 0$.
- 6: **nrhs** – Integer *Input*
On entry: r , the number of right-hand sides.
Constraint: **nrhs** ≥ 0 .
- 7: **a**[dim] – const Complex *Input*
Note: the dimension, dim , of the array **a** must be at least $\max(1, pda \times n)$.
On entry: the n by n triangular matrix A . If **uplo** = **Nag_Upper**, A is upper triangular and the elements of the array below the diagonal are not referenced; if **uplo** = **Nag_Lower**, A is lower triangular and the elements of the array above the diagonal are not referenced. If **diag** = **Nag_UnitDiag**, the diagonal elements of A are not referenced, but are assumed to be 1.
- 8: **pda** – Integer *Input*
On entry: the stride separating row or column elements (depending on the value of **order**) of the matrix A in the array **a**.
Constraint: **pda** $\geq \max(1, n)$.
- 9: **b**[dim] – Complex *Input/Output*
Note: the dimension, dim , of the array **b** must be at least $\max(1, pdb \times nrhs)$ when **order** = **Nag_ColMajor** and at least $\max(1, pdb \times n)$ when **order** = **Nag_RowMajor**.
 If **order** = **Nag_ColMajor**, the (i, j) th element of the matrix B is stored in **b**[($j - 1$) $\times$ **pdb** + $i - 1$] and if **order** = **Nag_RowMajor**, the (i, j) th element of the matrix B is stored in **b**[($i - 1$) $\times$ **pdb** + $j - 1$].
On entry: the n by r right-hand side matrix B .
On exit: the n by r solution matrix X .
- 10: **pdb** – Integer *Input*
On entry: the stride separating matrix row or column elements (depending on the value of **order**) in the array **b**.
Constraints:
 if **order** = **Nag_ColMajor**, **pdb** $\geq \max(1, n)$;
 if **order** = **Nag_RowMajor**, **pdb** $\geq \max(1, nrhs)$.
- 11: **fail** – NagError * *Output*
 The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **n** = $\langle value \rangle$.

Constraint: **n** ≥ 0 .

On entry, **nrhs** = $\langle value \rangle$.

Constraint: **nrhs** ≥ 0 .

On entry, **pda** = $\langle value \rangle$.

Constraint: **pda** > 0 .

On entry, **pdb** = $\langle value \rangle$.

Constraint: **pdb** > 0 .

NE_INT_2

On entry, **pda** = $\langle value \rangle$, **n** = $\langle value \rangle$.

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

On entry, **pdb** = $\langle value \rangle$, **n** = $\langle value \rangle$.

Constraint: **pdb** $\geq \max(1, \mathbf{n})$.

On entry, **pdb** = $\langle value \rangle$, **nrhs** = $\langle value \rangle$.

Constraint: **pdb** $\geq \max(1, \mathbf{nrhs})$.

NE_SINGULAR

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

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter $\langle value \rangle$ 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 solutions of triangular systems of equations are usually computed to high accuracy. See Higham (1989).

For each right-hand side vector b , the computed solution x is the exact solution of a perturbed system of equations $(A + E)x = b$, where

$$|E| \leq c(n)\epsilon|A|,$$

$c(n)$ is a modest linear function of n , and ϵ is the *machine precision*.

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x - \hat{x}\|_{\infty}}{\|x\|_{\infty}} \leq c(n) \operatorname{cond}(A, x)\epsilon,$$

provided $c(n) \operatorname{cond}(A, x)\epsilon < 1$, where $\operatorname{cond}(A, x) = \|A^{-1}\| |A| \|x\|_{\infty} / \|x\|_{\infty}$.

Note that $\operatorname{cond}(A, x) \leq \operatorname{cond}(A) = \|A^{-1}\| \|A\|_{\infty} \leq \kappa_{\infty}(A)$; $\operatorname{cond}(A, x)$ can be much smaller than $\operatorname{cond}(A)$ and it is also possible for $\operatorname{cond}(A^H)$, which is the same as $\operatorname{cond}(A^T)$, to be much larger (or smaller) than $\operatorname{cond}(A)$.

Forward and backward error bounds can be computed by calling `nag_ztrrfs` (f07tvc), and an estimate for $\kappa_{\infty}(A)$ can be obtained by calling `nag_ztrcon` (f07tuc) with `norm = Nag_InfNorm`.

8 Further Comments

The total number of real floating-point operations is approximately $4n^2r$.

The real analogue of this function is `nag_dtrtrs` (f07tec).

9 Example

To solve the system of equations $AX = B$, where

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

and

$$B = \begin{pmatrix} -14.78 - 32.36i & -18.02 + 28.46i \\ 2.98 - 2.14i & 14.22 + 15.42i \\ -20.96 + 17.06i & 5.62 + 35.89i \\ 9.54 + 9.91i & -16.46 - 1.73i \end{pmatrix}.$$

9.1 Program Text

```
/* nag_ztrtrs (f07tsc) 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 i, j, n, nrhs, pda, pdb;
    Integer exit_status=0;
    Nag_UploType uplo_enum;

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

#ifdef NAG_COLUMN_MAJOR
#define A(I,J) a[(J-1)*pda + I - 1]
#define B(I,J) b[(J-1)*pdb + I - 1]
    order = Nag_ColMajor;
#else
#define A(I,J) a[(I-1)*pda + J - 1]
#define B(I,J) b[(I-1)*pdb + J - 1]
    order = Nag_RowMajor;
#endif

    INIT_FAIL(fail);
    Vprintf("f07tsc Example Program Results\n\n");

    /* Skip heading in data file */
```

```

Vscanf("%*[^\\n] ");
Vscanf("%ld%ld%*[^\\n] ", &n, &nrhs);
#ifdef NAG_COLUMN_MAJOR
    pda = n;
    pdb = n;
#else
    pda = n;
    pdb = nrhs;
#endif

/* Allocate memory */
if ( !(a = NAG_ALLOC(n * n, Complex)) ||
     !(b = NAG_ALLOC(n * nrhs, Complex)) )
{
    Vprintf("Allocation failure\\n");
    exit_status = -1;
    goto END;
}

/* Read A and B 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(i,j).re, &A(i,j).im);
        Vscanf("%*[^\\n] ");
    }
}
else
{
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= i; ++j)
            Vscanf(" ( %lf , %lf )", &A(i,j).re, &A(i,j).im);
        Vscanf("%*[^\\n] ");
    }
}
for (i = 1; i <= n; ++i)
{
    for (j = 1; j <= nrhs; ++j)
        Vscanf(" ( %lf , %lf )", &B(i,j).re, &B(i,j).im);
}
Vscanf("%*[^\\n] ");

/* Compute solution */
f07tsc(order, uplo_enum, Nag_NoTrans, Nag_NonUnitDiag, n,
        nrhs, a, pda, b, pdb, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07tsc.\\n%s\\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print solution */
x04dbc(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, nrhs,
        b, pdb, Nag_BracketForm, "%7.4f", "Solution(s)",
        Nag_IntegerLabels, 0, Nag_IntegerLabels, 0, 80, 0,
        0, &fail);

```

```

if (fail.code != NE_NOERROR)
{
    Vprintf("Error from x04dbc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

END:
if (a) NAG_FREE(a);
if (b) NAG_FREE(b);

return exit_status;
}

```

9.2 Program Data

f07tsc Example Program Data

```

4 2                                     :Values of N and NRHS
'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
(-14.78,-32.36) (-18.02, 28.46)
( 2.98, -2.14) ( 14.22, 15.42)
(-20.96, 17.06) ( 5.62, 35.89)
( 9.54, 9.91) (-16.46, -1.73)           :End of matrix B

```

9.3 Program Results

f07tsc Example Program Results

```

Solution(s)
1 1 2
1 (-5.0000,-2.0000) ( 1.0000, 5.0000)
2 (-3.0000,-1.0000) (-2.0000,-2.0000)
3 ( 2.0000, 1.0000) ( 3.0000, 4.0000)
4 ( 4.0000, 3.0000) ( 4.0000,-3.0000)

```
