1 Purpose
nag_ztptrs (f07usc) solves a complex triangular system of linear equations with multiple right-hand sides, 
\( AX = B, A^T X = B \) or \( A^H X = B \), using packed storage.

2 Specification
void nag_ztptrs (Nag_OrderType order, Nag_UploType uplo, Nag_TransType trans,
Nag_DiagType diag, Integer n, Integer nrhs, const Complex ap[], Complex b[],
Integer pdb, NagError *fail)

3 Description
nag_ztptrs (f07usc) solves a complex triangular system of linear equations \( AX = B, A^T X = B \) or \( A^H X = B \) using packed storage.

4 References

5 Parameters
1: order – Nag_OrderType 
   \textit{Input}
   \textit{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 
   \textit{order} = Nag_RowMajor. See Section 2.2.1.4 of the Essential Introduction for a more detailed 
   explanation of the use of this parameter.
   \textit{Constraint:} order = Nag_RowMajor or Nag_ColMajor.

2: uplo – Nag_UploType 
   \textit{Input}
   \textit{On entry:} indicates whether \( A \) is upper or lower triangular as follows:
   - if \textit{uplo} = Nag_Upper, \( A \) is upper triangular;
   - if \textit{uplo} = Nag_Lower, \( A \) is lower triangular.
   \textit{Constraint:} uplo = Nag_Upper or Nag_Lower.

3: trans – Nag_TransType 
   \textit{Input}
   \textit{On entry:} indicates the form of the equations as follows:
   - if \textit{trans} = Nag_NoTrans, the equations are of the form \( AX = B \);
   - if \textit{trans} = Nag_Trans, the equations are of the form \( A^T X = B \);
   - if \textit{trans} = Nag_ConjTrans, the equations are of the form \( A^H X = B \).
   \textit{Constraint:} trans = Nag_NoTrans, Nag_Trans or Nag_ConjTrans.
4: diag – Nag_DiagType

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.

5: n – Integer

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

Constraint: $n$ ≥ 0.

6: nrhs – Integer

On entry: $r$, the number of right-hand sides.

Constraint: nrhs ≥ 0.

7: ap[dim] – const Complex

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 parameters 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 $i \geq j$;

if order = Nag_RowMajor and uplo = Nag_Lower, $a_{ij}$ is stored in $ap[(i - 1) \times i/2 + j - 1]$, for $i \geq j$.

8: b[dim] – Complex

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 $X$.

9: pdb – Integer

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 ≥ max(1, $n$);

if order = Nag_RowMajor, pdb ≥ max(1, nrhs).

10: 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: \( n \geq 0 \).

On entry, \( \text{nrhs} = \langle \text{value} \rangle \).
Constraint: \( \text{nrhs} \geq 0 \).

On entry, \( \text{pdb} = \langle \text{value} \rangle \).
Constraint: \( \text{pdb} > 0 \).

**NE_INT_2**

On entry, \( \text{pdb} = \langle \text{value} \rangle, n = \langle \text{value} \rangle \).
Constraint: \( \text{pdb} \geq \max(1, n) \).

On entry, \( \text{pdb} = \langle \text{value} \rangle, \text{nrhs} = \langle \text{value} \rangle \).
Constraint: \( \text{pdb} \geq \max(1, \text{nrhs}) \).

**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 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
\[
|E| / c(n) \leq |A| \epsilon \|x\|_{\infty} / \|x\|_{\infty},
\]
where \( c(n) \) is a modest linear function of \( n \), and \( \epsilon \) is the *machine precision*.

If \( \hat{x} \) is the true solution, then the computed solution \( x \) satisfies a forward error bound of the form
\[
\frac{\|x - \hat{x}\|_{\infty}}{\|x\|_{\infty}} \leq c(n) \|A\| \epsilon \|x\|_{\infty} / \|x\|_{\infty},
\]
where \( \text{cond}(A, x) = \|A^{-1}\| \|A\| \|x\|_{\infty} / \|x\|_{\infty} \).

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

Forward and backward error bounds can be computed by calling *nag_ztrrfs* (*f07ufc*), and an estimate for \( \kappa_\infty(A) \) can be obtained by calling *nag_ztpcon* (*f07uec*) with \( \text{norm} = \text{Nag Inf Norm} \).
8 Further Comments

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

The real analogue of this function is nag_dtptrs (f07uec).

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

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

using packed storage for $A$.

9.1 Program Text

/* nag_ztpttrs (f07usc) 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, nrhs, pdb;
    Integer exit_status=0;
    Nag_UploType uplo_enum;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    char uplo[2];
    Complex *ap=0, *b=0;

    INIT_FAIL(fail);
    Vprintf("f07usc Example Program Results\n\n");

    /* Skip heading in data file */

    #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]
    #define B(I,J) b[(J-1)*pdb + 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]
    #define B(I,J) b[(I-1)*pdb + J - 1]
    order = Nag_RowMajor;
    #endif

    INIT_FAIL(fail);
    Vprintf("f07usc Example Program Results\n\n");

    /* Skip heading in data file */
Vscanf("%*[\n]");
Vscanf("%ld%ld%*[\n] ", &n, &nrhs);
ap_len = n * (n + 1)/2;
#endif
#define Nag_Column_MAJOR
pdb = n;
#else
pdb = nrhs;
#endif

/* Allocate memory */
if ( !(ap = NAG_ALLOC(ap_len, 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;
    }

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] ");
    }

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] ");
    }

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

/* Compute solution */
f07usc(order, uplo_enum, Nag_NoTrans, Nag_NonUnitDiag, n,
nrhs, ap, b, pdb, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07usc.\n\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)
9.2 Program Data

f07usc 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

f07usc Example Program Results

Solution(s)

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)