NAG Library Function Document

nag_ztptrs (f07usc)

1 Purpose

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

2 Specification

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

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 \( AX = B \), \( A^TX = B \) or \( A^HX = B \), using packed storage.

4 References


5 Arguments

1: \( \text{order} \) – Nag_OrderType

\textit{Input}

\textit{On entry}: the \textit{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 \textit{order} = Nag_RowMajor. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

\textit{Constraint}: \( \text{order} \) = Nag_RowMajor or Nag_ColMajor.

2: \( \text{uplo} \) – Nag_UploType

\textit{Input}

\textit{On entry}: specifies whether \( A \) is upper or lower triangular.

\( \text{uplo} = \text{Nag}_\text{Upper} \)

\( A \) is upper triangular.

\( \text{uplo} = \text{Nag}_\text{Lower} \)

\( A \) is lower triangular.

\textit{Constraint}: \( \text{uplo} \) = Nag_Upper or Nag_Lower.

3: \( \text{trans} \) – Nag_TransType

\textit{Input}

\textit{On entry}: indicates the form of the equations.

\( \text{trans} = \text{Nag}_\text{NoTrans} \)

The equations are of the form \( AX = B \).
The equations are of the form $A^T X = B$.

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 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:** $n \geq 0$.

6: nrhs – Integer

**Input**

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

**Constraint:** nrhs $\geq 0$.

7: ap[dim] – const Complex

**Input**

*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 $a[i \times j/2 + i - 1]$, for $i \leq j$;

if order = Nag_ColMajor and uplo = Nag_Lower,

$A_{ij}$ is stored in $a[(j-1) \times (j-1)/2 + i - 1]$, for $i \geq j$;

if order = Nag_RowMajor and uplo = Nag_Upper,

$A_{ij}$ is stored in $a[i \times (i-1)/2 + j - 1]$, for $i \leq j$;

if order = Nag_RowMajor and uplo = Nag_Lower,

$A_{ij}$ is stored in $a[(i-1) \times i/2 + j - 1]$, 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.

8: 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;

$\max(1, n \times pdb)$ when order = Nag_RowMajor.

The $(i,j)$th element of the matrix $B$ is stored in

$b[(j-1) \times pdb + i - 1]$ when order = Nag_ColMajor;

$b[(i-1) \times pdb + j - 1]$ when order = Nag_RowMajor.

*On entry:* the $n$ by $r$ right-hand side matrix $B$.

*On exit:* the $n$ by $r$ solution matrix $X$. 

---

**f07usc**

*NAG Library Manual* 

Mark 25
9: \[ \textbf{pdb} \text{ – Integer} \quad \text{Input} \]

\textit{On entry:} the stride separating row or column elements (depending on the value of \textbf{order}) in the array \textbf{b}.

\textit{Constraints:}

\begin{align*}
\text{if order} & = \text{Nag\_ColMajor,} \quad \textbf{pdb} \geq \max(1, n); \\
\text{if order} & = \text{Nag\_RowMajor,} \quad \textbf{pdb} \geq \max(1, \textbf{nrhs}).
\end{align*}

10: \[ \textbf{fail} \text{ – NagError *} \quad \text{Input/Output} \]

\text{The NAG error argument (see Section 3.6 in the Essential Introduction).}

6 \quad \textbf{Error Indicators and Warnings}

\textbf{NE\_ALLOC\_FAIL}

Dynamic memory allocation failed.

See Section 3.2.1.2 in the Essential Introduction for further information.

\textbf{NE\_BAD\_PARAM}

\textit{On entry, argument \langle value \rangle had an illegal value.}

\textbf{NE\_INT}

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

\text{Constraint:} \( n \geq 0 \).

\textit{On entry,} \( \textbf{nrhs} = \langle value \rangle \).

\text{Constraint:} \( \textbf{nrhs} \geq 0 \).

\textit{On entry,} \( \textbf{pdb} = \langle value \rangle \).

\text{Constraint:} \( \textbf{pdb} > 0 \).

\textbf{NE\_INT\_2}

\textit{On entry,} \( \textbf{pdb} = \langle value \rangle \) and \( n = \langle value \rangle \).

\text{Constraint:} \( \textbf{pdb} \geq \max(1, n) \).

\textit{On entry,} \( \textbf{pdb} = \langle value \rangle \) and \( \textbf{nrhs} = \langle value \rangle \).

\text{Constraint:} \( \textbf{pdb} \geq \max(1, \textbf{nrhs}) \).

\textbf{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.

\textbf{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.

\textbf{NE\_SINGULAR}

Element \( \langle value \rangle \) of the diagonal is exactly zero. \( A \) is singular and the solution has not been computed.
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

\[
|E| \leq c(n)|A|,
\]

\( c(n) \) 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

\[
\frac{||x - \hat{x}||_\infty}{||x||_\infty} \leq c(n) \text{cond}(A, x)\epsilon, \quad \text{provided} \quad c(n) \text{cond}(A, x)\epsilon < 1,
\]

where \( \text{cond}(A, x) = \frac{||A^{-1}||_\infty||x||_\infty}{||x||_\infty} \).

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

Forward and backward error bounds can be computed by calling \texttt{nag_ztprfs} (f07uvc), and an estimate for \( \kappa_\infty(A) \) can be obtained by calling \texttt{nag_ztpcon} (f07uuc) with \( \text{norm} = \text{Nag}_{\text{InfNorm}} \).

8 Parallelism and Performance

\texttt{nag_ztptrs} (f07usc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

\texttt{nag_ztptrs} (f07usc) 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 \( 4n^2r \).

The real analogue of this function is \texttt{nag_dtptrs} (f07uec).

10 Example

This example solves 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 \( A \).
10.1 Program Text

/* nag_ztptrs (f07usc) Example Program.  
* Copyright 2014 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, nrhs, pdb;
  Integer exit_status = 0;
  Nag_UploType uplo;
  NagError fail;
  Nag_OrderType order;
  /* Arrays */
  char nag_enum_arg[40];
  Complex *ap = 0, *b = 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]
  #define B(I, J) b[(J-1)*pdb + I - 1]
  #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]
  #endif

  INIT_FAIL(fail);
  printf("nag_ztptrs (f07usc) 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 "%"NAG_IFMT "%*\n", &n, &nrhs);
  #else
  scanf("%"NAG_IFMT "%"NAG_IFMT "%*\n", &n, &nrhs);
  #endif
  ap_len = n * (n + 1)/2;
  #ifdef NAG_COLUMN_MAJOR
  pdb = n;
  #else
  pdb = nrhs;
  #endif

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

/* Read A and B 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
        for (i = 1; i <= nrhs; ++i)
        {
            for (j = 1; j <= nrhs; ++j)
                #ifdef _WIN32
                    scanf_s(" ( %lf , %lf )", &B(i, j).re, &B(i, j).im);
                #else
                    scanf(" ( %lf , %lf )", &B(i, j).re, &B(i, j).im);
                #endif
        }
        #ifdef _WIN32
            scanf_s("%\n ");
        #else
            scanf("%\n ");
        #endif
        /* Compute solution */
/* nag_ztptrs (f07usc).
   * Solution of complex triangular system of linear
   * equations, multiple right-hand sides, packed storage
   */
nag_ztptrs(order, uplo, Nag_NoTrans, Nag_NonUnitDiag, n,
            nrhs, ap, b, pdb, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_ztptrs (f07usc).\n\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print solution */
/* nag_gen_complx_mat_print_comp (x04dbc).
   * Print complex general matrix (comprehensive)
   */
fflush(stdout);
nag_gen_complx_mat_print_comp(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)
{
    printf("Error from nag_gen_complx_mat_print_comp (x04dbc).\n\n", fail.message);
    exit_status = 1;
    goto END;
}

END:
NAG_FREE(ap);
NAG_FREE(b);
return exit_status;

10.2 Program Data

nag_ztptrs (f07usc) Example Program Data

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Nag_Lower  :Values of n and nrhs
|    |    |    |    |
|  ( 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) | ( 0.33,-0.26) |
|    |    |    |    |
| ( 2.98, -2.14) | ( 14.22, 15.42) |
|    |    |    |    |
| (-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) |
|    |    |    |    |

:Value of uplo

:End of matrix A

:End of matrix B

10.3 Program Results

nag_ztptrs (f07usc) Example Program Results

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Solution(s)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(-5.0000,-2.0000)</td>
<td>( 1.0000, 5.0000)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(-3.0000,-1.0000)</td>
<td>(-2.0000,-2.0000)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 2.0000, 1.0000)</td>
<td>( 3.0000, 4.0000)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 4.0000, 3.0000)</td>
<td>( 4.0000,-3.0000)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>