NAG 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

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

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


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: uplo – Nag_UploType

On entry: specifies whether $A$ is upper or lower triangular.

$uplo = \text{Nag_Upper}$

$A$ is upper triangular.

$uplo = \text{Nag_Lower}$

$A$ is lower triangular.

Constraint: uplo = Nag_Upper or Nag_Lower.

3: trans – Nag_TransType

On entry: indicates the form of the equations.

$trans = \text{Nag_NoTrans}$

The equations are of the form $AX = B$. 
\texttt{trans} = \texttt{Nag\_Trans} \\
\hspace{1em} The equations are of the form $A^T X = B$.

\texttt{trans} = \texttt{Nag\_ ConjTrans} \\
\hspace{1em} The equations are of the form $A^H X = B$.

\textit{Constraint:} \texttt{trans} = \texttt{Nag\_NoTrans}, \texttt{Nag\_Trans} or \texttt{Nag\_ConjTrans}.

4: \hspace{0.5em} \texttt{diag} = \texttt{Nag\_DiagType} \hspace{1em} \textit{Input}

\textit{On entry:} indicates whether $A$ is a nonunit or unit triangular matrix.

\texttt{diag} = \texttt{Nag\_NonUnitDiag} \\
\hspace{1em} $A$ is a nonunit triangular matrix.

\texttt{diag} = \texttt{Nag\_UnitDiag} \\
\hspace{1em} $A$ is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

\textit{Constraint:} \texttt{diag} = \texttt{Nag\_NonUnitDiag} or \texttt{Nag\_UnitDiag}.

5: \hspace{0.5em} \texttt{n} = Integer \hspace{1em} \textit{Input}

\textit{On entry:} $n$, the order of the matrix $A$.

\textit{Constraint:} \texttt{n} \geq 0.

6: \hspace{0.5em} \texttt{nrhs} = Integer \hspace{1em} \textit{Input}

\textit{On entry:} $r$, the number of right-hand sides.

\textit{Constraint:} \texttt{nrhs} \geq 0.

7: \hspace{0.5em} \texttt{a[dim]} = \text{const Complex} \hspace{1em} \textit{Input}

\textit{Note:} the dimension, \texttt{dim}, of the array \texttt{a} must be at least \texttt{max(1, pda \times n)}.

\textit{On entry:} the $n$ by $n$ triangular matrix $A$.

If \texttt{order} = \texttt{Nag\_ColMajor}, $A_{ij}$ is stored in \texttt{a[(j - 1) \times pda + i - 1]}.

If \texttt{order} = \texttt{Nag\_RowMajor}, $A_{ij}$ is stored in \texttt{a[(i - 1) \times pda + j - 1]}.

If \texttt{uplo} = \texttt{Nag\_Upper}, the upper triangular part of $A$ must be stored and the elements of the array below the diagonal are not referenced.

If \texttt{uplo} = \texttt{Nag\_Lower}, the lower triangular part of $A$ must be stored and the elements of the array above the diagonal are not referenced.

If \texttt{diag} = \texttt{Nag\_UnitDiag}, the diagonal elements of $A$ are assumed to be 1, and are not referenced.

8: \hspace{0.5em} \texttt{pda} = Integer \hspace{1em} \textit{Input}

\textit{On entry:} the stride separating row or column elements (depending on the value of \texttt{order}) of the matrix $A$ in the array \texttt{a}.

\textit{Constraint:} \texttt{pda} \geq \texttt{max(1, n)}.

9: \hspace{0.5em} \texttt{b[dim]} = Complex \hspace{1em} \textit{Input/Output}

\textit{Note:} the dimension, \texttt{dim}, of the array \texttt{b} must be at least

\texttt{max(1, pdb \times nrhs)} when \texttt{order} = \texttt{Nag\_ColMajor};

\texttt{max(1, n \times pdb)} when \texttt{order} = \texttt{Nag\_RowMajor}.

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

- $b[(j - 1) \times pdb + i - 1]$ when \texttt{order} = \texttt{Nag\_ColMajor};
- $b[(i - 1) \times pdb + j - 1]$ when \texttt{order} = \texttt{Nag\_RowMajor}.
On entry: the $n$ by $r$ right-hand side matrix $B$.
On exit: the $n$ by $r$ solution matrix $X$.

10: pdb – Integer

On entry: the stride separating 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 *

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 $\langle\text{value}\rangle$ had an illegal value.

NE_INT
On entry, $n = \langle\text{value}\rangle$.
Constraint: $n \geq 0$.
On entry, $nrhs = \langle\text{value}\rangle$.
Constraint: $nrhs \geq 0$.
On entry, $pda = \langle\text{value}\rangle$.
Constraint: $pda > 0$.
On entry, $pdb = \langle\text{value}\rangle$.
Constraint: $pdb > 0$.

NE_INT_2
On entry, $pda = \langle\text{value}\rangle$ and $n = \langle\text{value}\rangle$.
Constraint: $pda \geq \max(1, n)$.
On entry, $pdb = \langle\text{value}\rangle$ and $n = \langle\text{value}\rangle$.
Constraint: $pdb \geq \max(1, n)$.
On entry, $pdb = \langle\text{value}\rangle$ and $nrhs = \langle\text{value}\rangle$.
Constraint: $pdb \geq \max(1, nrhs)$.

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.
Element $\langle \text{value} \rangle$ 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

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

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

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

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

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

Note that $\text{cond}(A,x) \leq \text{cond}(A) = \|||A^{-1}||A||_\infty \leq \kappa_\infty(A)$; $\text{cond}(A,x)$ can be much smaller than $\text{cond}(A)$ and it is also possible for $\text{cond}(A^H)$, 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 (f07tvc), and an estimate for $\kappa_\infty(A)$ can be obtained by calling nag_ztrecn (f07tuc) with norm = Nag_InfNorm.

8 Parallelism and Performance

nag_ztrtrs (f07tsc) is not threaded by NAG in any implementation.

nag_ztrtrs (f07tsc) 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 nag_dtrtrs (f07tec).

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

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

```c
/* nag_ztrtrs (f07tsc) 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 i, j, n, nrhs, pda, pdb;
    Integer exit_status = 0;
    Nag_UploType uplo;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    char nag_enum_arg[40];
    Complex *a = 0, *b = 0;

    if (!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(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);
    printf("nag_ztrtrs (f07tsc) 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

    if (!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)))
    {
        printf("Allocation failure\n");
    }
}
```

Mark 25
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(i, j).re, &A(i, j).im);
#else
            scanf(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
#endif
    }
#endif _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(i, j).re, &A(i, j).im);
#else
            scanf(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
#endif
    }
#endif _WIN32
    scanf_s("%[*\n] ");
#else
    scanf("%[*\n] ");
#endif
for (i = 1; i <= n; ++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
    }
#endif _WIN32
    scanf_s("%[*\n] ");
#else
    scanf("%[*\n] ");
#endif

/* Compute solution */
#ifndef _WIN32
    scanf(" %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 */
nrhs, a, pda, b, pdb, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_ztrtrs (f07tsc).\n%s\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_LogUnitDiag, 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%s\n",
            fail.message);
    exit_status = 1;
    goto END;
}

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

10.2 Program Data

nag_ztrtrs (f07tsc) Example Program Data

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>M</td>
</tr>
<tr>
<td></td>
<td>Values of n and nrhs</td>
<td>Values of uplo</td>
</tr>
<tr>
<td></td>
<td>( 4.78, 4.56)</td>
<td>( 4.11, 1.25)</td>
</tr>
<tr>
<td></td>
<td>( 2.00, 0.30)</td>
<td>(-14.78, -32.36)</td>
</tr>
<tr>
<td></td>
<td>( 2.89, -1.34)</td>
<td>(-18.02, 28.46)</td>
</tr>
<tr>
<td></td>
<td>( 1.89, 1.15)</td>
<td>( 14.22, 15.42)</td>
</tr>
<tr>
<td></td>
<td>(0.04, -3.69)</td>
<td>( 5.62, 35.89)</td>
</tr>
<tr>
<td></td>
<td>(-0.02, 0.46)</td>
<td>(-16.46, -1.73)</td>
</tr>
<tr>
<td></td>
<td>( 0.33, -0.26)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( 4.15, 0.80)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( 2.36, -4.25)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( 4.15, 0.80)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(-14.78, -32.36)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(-18.02, 28.46)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(-14.78, -32.36)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(-18.02, 28.46)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(-14.78, -32.36)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(-18.02, 28.46)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(-14.78, -32.36)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(-18.02, 28.46)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(-14.78, -32.36)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(-18.02, 28.46)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(-14.78, -32.36)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(-18.02, 28.46)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(-14.78, -32.36)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(-18.02, 28.46)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(-14.78, -32.36)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(-18.02, 28.46)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(-14.78, -32.36)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(-18.02, 28.46)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(-14.78, -32.36)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(-18.02, 28.46)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(-14.78, -32.36)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(-18.02, 28.46)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(-14.78, -32.36)</td>
<td></td>
</tr>
</tbody>
</table>

10.3 Program Results

nag_ztrtrs (f07tsc) Example Program Results

Solution(s)

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