NAG Library Function Document

nag_ztrsm (f16zjc)

1 Purpose

nag_ztrsm (f16zjc) solves a system of equations given as a complex triangular matrix with multiple right-hand sides.

2 Specification

```c
#include <nag.h>
#include <nagf16.h>

void nag_ztrsm (Nag_OrderType order, Nag_SideType side, Nag_UploType uplo,
               Nag_TransType trans, Nag_DiagType diag, Integer m, Integer n,
               Complex alpha, const Complex a[], Integer pda, Complex b[], Integer pdb,
               NagError *fail)
```

3 Description

nag_ztrsm (f16zjc) performs one of the matrix-matrix operations

\[ B \leftarrow \alpha A^{-1}B, \quad B \leftarrow \alpha A^{-\top}B, \quad B \leftarrow \alpha A^{-\mathbb{H}}B, \]

\[ B \leftarrow \alpha B A^{-1}, \quad B \leftarrow \alpha B A^{-\top} \text{ or } B \leftarrow \alpha B A^{-\mathbb{H}}, \]

where \( A \) is a complex triangular matrix, \( B \) is an \( m \) by \( n \) complex matrix, and \( \alpha \) is a complex scalar. \( A^{-\top} \) denotes \( A^{-\top} \) or equivalently \( A^{-\top} \); \( A^{-\mathbb{H}} \) denotes \( (A^{\mathbb{H}})^{-1} \) or equivalently \( (A^{-1})^{\mathbb{H}} \).

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:  
side – Nag_SideType

On entry: specifies whether \( B \) is operated on from the left or the right.

side = Nag_LeftSide
\( B \) is pre-multiplied from the left.

side = Nag_RightSide
\( B \) is post-multiplied from the right.

Constraint: side = Nag_LeftSide or Nag_RightSide.
3: 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.

4: trans – Nag_TransType  
   Input
   On entry: specifies the operation to be performed.
   trans = Nag_Trans and side = Nag_LeftSide
   $B \leftarrow \alpha A^{-T}B$.
   trans = Nag_NoTrans and side = Nag_LeftSide
   $B \leftarrow \alpha A^{-1}B$.
   trans = Nag_ConjTrans and side = Nag_LeftSide
   $B \leftarrow \alpha A^{-H}B$.
   trans = Nag_Trans and side = Nag_RightSide
   $B \leftarrow \alpha BA^{-T}$.
   trans = Nag_NoTrans and side = Nag_RightSide
   $B \leftarrow \alpha BA^{-1}$.
   trans = Nag_ConjTrans and side = Nag_RightSide
   $B \leftarrow \alpha BA^{-H}$.
   Constraints:
   side = Nag_LeftSide or Nag_RightSide;
   trans = Nag_NoTrans or Nag_Trans.

5: diag – Nag_DiagType  
   Input
   On entry: specifies whether $A$ has nonunit or unit diagonal elements.
   diag = Nag_NonUnitDiag
   The diagonal elements are stored explicitly.
   diag = Nag_UnitDiag
   The diagonal elements are assumed to be 1 and are not referenced.
   Constraint: diag = Nag_NonUnitDiag or Nag_UnitDiag.

6: m – Integer  
   Input
   On entry: $m$, the number of rows of the matrix $B$; the order of $A$ if side = Nag_LeftSide.
   Constraint: $m \geq 0$.

7: n – Integer  
   Input
   On entry: $n$, the number of columns of the matrix $B$; the order of $A$ if side = Nag_RightSide.
   Constraint: $n \geq 0$.

8: alpha – Complex  
   Input
   On entry: the scalar $\alpha$. 
9:  
   \(a[dim]\) – const Complex

   **Note:** the dimension, \(dim\), of the array \(a\) must be at least
   \(\max(1, pda \times m)\) when \(side = \text{Nag\_LeftSide}\);
   \(\max(1, pda \times n)\) when \(side = \text{Nag\_RightSide}\).

   On entry: the triangular matrix \(A\); \(A\) is \(m\) by \(m\) if \(side = \text{Nag\_LeftSide}\), or \(n\) by \(n\) if \(side = \text{Nag\_RightSide}\).

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

   If \(uplo = \text{Nag\_Upper}\), \(A\) is upper triangular and the elements of the array corresponding to the lower triangular part of \(A\) are not referenced.
   If \(uplo = \text{Nag\_Lower}\), \(A\) is lower triangular and the elements of the array corresponding to the upper triangular part of \(A\) are not referenced.
   If \(diag = \text{Nag\_UnitDiag}\), the diagonal elements of \(A\) are assumed to be 1, and are not referenced.

10:  
   \(pda\) – Integer

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

   Constraints:
   
   if \(side = \text{Nag\_LeftSide}\), \(pda \geq \max(1, m)\);
   if \(side = \text{Nag\_RightSide}\), \(pda \geq \max(1, n)\).

11:  
   \(b[dim]\) – Complex

   **Note:** the dimension, \(dim\), of the array \(b\) must be at least
   \(\max(1, pdb \times n)\) when \(order = \text{Nag\_ColMajor}\);
   \(\max(1, m \times pdb)\) when \(order = \text{Nag\_RowMajor}\).

   If \(order = \text{Nag\_ColMajor}\), \(B_{ij}\) is stored in \(b[(j-1) \times pdb + i - 1]\).
   If \(order = \text{Nag\_RowMajor}\), \(B_{ij}\) is stored in \(b[(i-1) \times pdb + j - 1]\).

   On entry: the \(m\) by \(n\) matrix \(B\).

   If \(alpha = 0\), \(b\) need not be set.

   On exit: the updated matrix \(B\).

12:  
   \(pdb\) – Integer

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

   Constraints:
   
   if \(order = \text{Nag\_ColMajor}\), \(pdb \geq \max(1, m)\);
   if \(order = \text{Nag\_RowMajor}\), \(pdb \geq \max(1, n)\).

13:  
   \(fail\) – 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 <em>value</em> had an illegal value.

**NE_ENUM_INT_2**

On entry, <em>side</em> = <em>value</em>, <em>pda</em> = <em>value</em>, <em>m</em> = <em>value</em>.

Constraint: if <em>side</em> = Nag_LeftSide, <em>pda</em> ≥ max(1, <em>m</em>).

On entry, <em>side</em> = <em>value</em>, <em>pda</em> = <em>value</em>, <em>n</em> = <em>value</em>.

Constraint: if <em>side</em> = Nag_RightSide, <em>pda</em> ≥ max(1, <em>n</em>).

**NE_INT**

On entry, <em>m</em> = <em>value</em>.

Constraint: <em>m</em> ≥ 0.

On entry, <em>n</em> = <em>value</em>.

Constraint: <em>n</em> ≥ 0.

**NE_INT_2**

On entry, <em>pda</em> = <em>value</em>; <em>n</em> = <em>value</em>.

Constraint: <em>pda</em> ≥ max(1, <em>n</em>).

On entry, <em>pdb</em> = <em>value</em>; <em>m</em> = <em>value</em>.

Constraint: <em>pdb</em> ≥ max(1, <em>m</em>).

On entry, <em>pdb</em> = <em>value</em>, <em>m</em> = <em>value</em>.

Constraint: <em>pdb</em> ≥ max(1, <em>m</em>).

On entry, <em>pdb</em> = <em>value</em> and <em>n</em> = <em>value</em>.

Constraint: <em>pdb</em> ≥ max(1, <em>n</em>).

**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.

### 7 Accuracy

The BLAS standard requires accurate implementations which avoid unnecessary over/underflow (see Section 2.7 of Basic Linear Algebra Subprograms Technical (BLAST) Forum (2001)).

### 8 Parallelism and Performance

`nag_ztrsm (f16zjc)` is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

`nag_ztrsm (f16zjc)` 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

No test for singularity or near-singularity of \( A \) is included in nag_ztrsm (f16zjc). Such tests must be performed before calling this function.

10 Example

Premultiply complex 4 by 2 matrix \( B \) by inverse of lower triangular 4 by 4 matrix \( A \), \( B \leftarrow A^{-1}B \) (or solve \( AX = B \) and return result in \( B \)), where

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

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

/* nag_ztrsm (f16zjc) Example Program. *
* Copyright 2014 Numerical Algorithms Group. *
* Mark 8, 2005. */

#include <stdio.h>
#include <nag.h>
#include <nagf16.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    Complex alpha;
    Integer exit_status, i, j, m, n, pda, pdb;

    /* Arrays */
    Complex *a = 0, *b = 0;
    char nag_enum_arg[40];

    /* Nag Types */
    NagError fail;
    Nag_SideType side;
    Nag_DiagType diag;
    Nag_OrderType order;
    Nag_TransType trans;
    Nag_UploType uplo;

    /* Scalars */
    Complex alpha;
    Integer exit_status, i, j, m, n, pda, pdb;

    /* Arrays */
    Complex *a = 0, *b = 0;
    char nag_enum_arg[40];

    /* Nag Types */
    NagError fail;
    Nag_SideType side;
    Nag_DiagType diag;
    Nag_OrderType order;
    Nag_TransType trans;
    Nag_UploType uplo;

    #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

    exit_status = 0;
INIT_FAIL(fail);

printf("nag_ztrsm (f16zjc) Example Program Results\n\n");

/* Skip heading in data file */
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif
/* Read the problem dimensions */
#ifdef _WIN32
    scanf_s("%"NAG_IFMT"%"NAG_IFMT"%*[\n] ", &m, &n);
#else
    scanf("%"NAG_IFMT"%"NAG_IFMT"%*[\n] ", &m, &n);
#endif
#ifdef NAG_COLUMN_MAJOR
    pdb = m;
#else
    pdb = n;
#endif

/* Read side */
#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 */
side = (Nag_SideType) nag_enum_name_to_value(nag_enum_arg);

/* Read uplo */
#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), see above. */
uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);

/* Read trans */
#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), see above. */
trans = (Nag_TransType) nag_enum_name_to_value(nag_enum_arg);

/* Read diag */
#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), see above. */
diag = (Nag_DiagType) nag_enum_name_to_value(nag_enum_arg);

/* Read scalar parameters */
#ifdef _WIN32
    scanf_s(" ( %lf , %lf )%*[\n] ", &alpha.re, &alpha.im);
#else
    scanf(" ( %lf , %lf )%*[\n] ", &alpha.re, &alpha.im);
#endif
if (side == Nag_LeftSide)
    { pda = m; }
else
    { pda = n; }
if (n > 0)
{
    /* Allocate memory */
    if (!((a = NAG_ALLOC(pda*pda, Complex)) || !((b = NAG_ALLOC(n*m, Complex)))))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
}
else
{
    printf("Invalid n\n");
    exit_status = 1;
    return exit_status;
}

/* Read A from data file */
if (uplo == Nag_Upper)
{
    for (i = 1; i <= pda; ++i)
    {
        for (j = i; j <= pda; ++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
        }
#ifdef _WIN32
        scanf_s("%*['\n "");
#else
        scanf("%*['\n "");
#endif
    }
}
else
{
    for (i = 1; i <= pda; ++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
        }
#ifdef _WIN32
        scanf_s("%*['\n "");
#else
        scanf("%*['\n "");
#endif
    }
}

/* Input matrix B */
for (i = 1; i <= m; ++i)
{
    for (j = 1; j <= n; ++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
    }
}

/* nag_ztrsm (f16zjc).
 * Multiply matrix by inverse of Triangular complex matrix.
 */
nag_ztrsm(order, side, uplo, trans, diag, m, n, alpha, a, pda, b, pdb, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_ztrsm (f16zjc).\n\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Print the updated matrix B */
/* nag_gen_complx_mat_print_comp (x04dbc). */
/* Print complex general matrix (comprehensive) */
fflush(stdout);
nag_gen_complx_mat_print_comp(order, Nag_GeneralMatrix, Nag_NoTrans, Nag_Lower, Nag_NonUnitDiag, m, n, b, pdb, Nag_BracketForm, "%5.1f", "Updated Matrix B", 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(a);
NAG_FREE(b);
return exit_status;

10.2 Program Data
nag_ztrsm (f16zjc) Example Program Data
4 2 :Values of m and n
   Nag_LeftSide :Value of side
   Nag_Lower :Value of uplo
   Nag_NoTrans :Value of trans
   Nag_NonUnitDiag :Value of diag
( 1.00, 0.00) :Value of alpha
( 4.78, 4.56)
( 2.00,-0.30) (-4.11, 1.25)
( 2.89,-1.34) ( 2.00,-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

10.3 Program Results
nag_ztrsm (f16zjc) Example Program Results
Updated Matrix B
1 2
1 (-5.0, -2.0) ( 1.0, 5.0)
2 (-3.0, -1.0) (-2.0, -2.0)
3 ( 2.0, 1.0) ( 3.0, 4.0)
4 ( 4.0, 3.0) ( 4.0, -3.0)