NAG C Library Function Document

nag_ztrsm (f16zjc)

1 Purpose

nag_ztrsm (f16zjc) solves one of the matrix equations

$$TX = \alpha B, \quad T^T X = \alpha B, \quad T^H X = \alpha B, \quad XT = \alpha B, \quad XT^T = \alpha B \quad \text{or} \quad XT^H = \alpha B,$$

where X and B are m by n complex matrices and T is a complex triangular matrix.

2 Specification

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

3 Description

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

$$\begin{array}{lll} B \leftarrow \alpha T^{-1} B, & B \leftarrow \alpha T^{-T} B, & B \leftarrow \alpha T^{-H} B, \\ B \leftarrow \alpha B T^{-1}, & B \leftarrow \alpha B T^{-T} \quad \text{or} & B \leftarrow \alpha B T^{-H}, \end{array}$$

where T is a complex triangular matrix, B is an m by n complex matrix, and α is a complex scalar. T^{-T} denotes $(T^T)^{-1}$ or equivalently $(T^{-1})^T$; T^{-H} denotes $(T^H)^{-1}$ or equivalently $(T^{-1})^H$.

4 References

The BLAS Technical Forum Standard (2001) *www.netlib.org/blas/blast-forum*

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: **side** – Nag_SideType *Input*

On entry: specifies whether B is operated on from the left or the right, as follows:

if **side** = **Nag_LeftSide**, B is pre-multiplied from the left;

if **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 T is upper or lower triangular as follows:

if **uplo** = **Nag_Upper**, T is upper triangular;

if **uplo** = **Nag_Lower**, T is lower triangular.

Constraint: **uplo** = **Nag_Upper** or **Nag_Lower**.

4: **transt** – Nag_TransType *Input*

On entry: specifies the operation to be performed as follows:

if **side** = **Nag_LeftSide** and **transt** = **Nag_Trans**, $B \leftarrow \alpha T^{-T} B$;

if **side** = **Nag_LeftSide** and **transt** = **Nag_NoTrans**, $b \leftarrow \alpha T^{-1} B$;

if **side** = **Nag_LeftSide** and **transt** = **Nag_ConjTrans**, $B \leftarrow \alpha T^{-H} B$;

if **side** = **Nag_RightSide** and **transt** = **Nag_Trans**, $B \leftarrow \alpha B T^{-T}$;

if **side** = **Nag_RightSide** and **transt** = **Nag_NoTrans**, $B \leftarrow \alpha B T^{-1}$.

if **side** = **Nag_RightSide** and **transt** = **Nag_ConjTrans**, $B \leftarrow \alpha B T^{-H}$.

Constraint: **side** = **Nag_LeftSide** or **Nag_RightSide**; **transt** = **Nag_NoTrans** or **Nag_Trans**.

5: **diag** – Nag_DiagType *Input*

On entry: specifies whether A has non-unit or unit diagonal elements, as follows:

if **diag** = **Nag_NonUnitDiag**, the diagonal elements are stored explicitly;

if **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 T if **side** = **Nag_LeftSide**.

Constraint: **m** ≥ 0.

7: **n** – Integer *Input*

On entry: n, the number of columns of the matrix B; the order of T if **side** = **Nag_RightSide**.

Constraint: **n** ≥ 0.

8: **alpha** – Complex *Input*

On entry: the scalar α.

9: **t**[dim] – const Complex *Input*

Note: the dimension, dim, of the array **t** must be at least $\max(1, \textbf{pdt} \times \textbf{m})$ when **side** = **Nag_LeftSide** and at least $\max(1, \textbf{pdt} \times \textbf{n})$ when **side** = **Nag_RightSide**.

If **order** = **Nag_ColMajor**, the (i,j)th element of the matrix T is stored in **t**$[(j-1) \times \textbf{pdt} + i - 1]$ and if **order** = **Nag_RowMajor**, the (i,j)th element of the matrix T is stored in **t**$[(i-1) \times \textbf{pdt} + j - 1]$.

On entry: the m by m triangular matrix T if **side** = **Nag_LeftSide** or n by n triangular matrix T if **side** = **Nag_RightSide**. If **uplo** = **Nag_Upper**, T is upper triangular and the elements of the array below the diagonal are not referenced; if **uplo** = **Nag_Lower**, T is lower triangular and the elements of the array above the diagonal are not referenced. If **diag** = **Nag_UnitDiag**, the diagonal elements of T are not referenced, but are assumed to be 1.

10: **pdt** – Integer *Input*

On entry: the stride separating row or column elements (depending on the value of **order**) of the matrix T in the array **t**.

Constraints:

if **side** = **Nag_LeftSide**, **pdt** $\geq \max(1, \textbf{m})$;
if **side** = **Nag_RightSide**, **pdt** $\geq \max(1, \textbf{n})$.

11: **b**[dim] – Complex *Input/Output*

Note: the dimension, dim, of the array **b** must be at least $\max(1, \mathbf{pdb} \times \mathbf{n})$ when **order** = **Nag_ColMajor** and at least $\max(1, \mathbf{pdb} \times \mathbf{m})$ when **order** = **Nag_RowMajor**.

If **order** = **Nag_ColMajor**, the (i, j)th element of the matrix B is stored in **b**$[(j-1) \times \mathbf{pdb} + i - 1]$ and if **order** = **Nag_RowMajor**, the (i, j)th element of the matrix B is stored in **b**$[(i-1) \times \mathbf{pdb} + j - 1]$.

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 *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** $\geq \max(1, \mathbf{m})$;
if **order** = **Nag_RowMajor**, **pdb** $\geq \max(1, \mathbf{n})$.

13: **fail** – NagError * *Input/Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **m** = ⟨*value*⟩.
Constraint: **m** ≥ 0.

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 0.

On entry, **pdt** = ⟨*value*⟩.
Constraint: **pdt** $\geq \max(1, \mathbf{n})$.

On entry, **pdb** = ⟨*value*⟩.
Constraint: **pdb** $\geq \max(1, \mathbf{m})$.

On entry, **pdb** = ⟨*value*⟩.
Constraint: **pdb** $\geq \max(1, \mathbf{n})$.

NE_BAD_PARAM

On entry, parameter ⟨*value*⟩ had an illegal value.

7 Accuracy

The BLAS standard requires accurate implementations which avoid unnecessary over/underflow (see section 2.7 of The BLAS Technical Forum Standard (2001)).

8 Further Comments

No test for singularity or near-singularity of T is included in this routine. Such tests must be performed before calling this routine.

9 Example

None.