NAG Library Function Document

nag_ztr_copy (f16tec)

1 Purpose

nag_ztr_copy (f16tec) copies a complex triangular matrix.

2 Specification

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

void nag_ztr_copy (Nag_OrderType order, Nag_UploType uplo, Nag_TransType trans, Nag_DiagType diag, Integer n, const Complex a[], Integer pda, Complex b[], Integer pdb, NagError *fail)

3 Description

nag_ztr_copy (f16tec) performs the triangular matrix copy operations

\[
B \leftarrow A, \quad B \leftarrow A^T \quad \text{or} \quad B \leftarrow A^H
\]

where \( A \) and \( B \) are \( n \) by \( n \) complex triangular matrices.

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 the upper or lower triangular part of \( A \) is stored.
    
    uplo = Nag_Upper
    
    The upper triangular part of \( A \) is stored.
    
    uplo = Nag_Lower
    
    The lower triangular part of \( A \) is stored.
    
    Constraint: uplo = Nag_Upper or Nag_Lower.

3:  trans – Nag_TransType
    
    On entry: specifies the operation to be performed.
    
    trans = Nag_NoTrans
    
    \( B \leftarrow A \).
trans = Nag_Trans
    B ← Aᵀ.

trans = Nag_ConjTrans
    B ← Aᴴ.

Constraint: trans = Nag_NoTrans, Nag_Trans or Nag_ConjTrans.

4: 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.

5: n – Integer
    Input

On entry: n, the order of the matrices A and B.

Constraint: n ≥ 0.

6: a[dim] – const Complex
    Input

Note: the dimension, dim, of the array a must be at least max(1, pd a × n).

On entry: the n by n triangular matrix A.

If order = Nag_ColMajor, Aᵢⱼ is stored in a[(j - 1) × pd a + i - 1].

If order = Nag_RowMajor, Aᵢⱼ is stored in a[(i - 1) × pd a + j - 1].

If uplo = Nag_Upper, the upper triangular part of A must be stored and the elements of the array below the diagonal are not referenced.

If uplo = Nag_Lower, the lower triangular part of A must be stored and the elements of the array above the diagonal are not referenced.

If diag = Nag_UnitDiag, the diagonal elements of A are assumed to be 1, and are not referenced.

7: pda – Integer
    Input

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

Constraint: pda ≥ max(1, n).

8: b[dim] – Complex
    Output

Note: the dimension, dim, of the array b must be at least max(1, pd b × n).

On exit: the n by n triangular matrix B.

If order = Nag_ColMajor, Bᵢⱼ is stored in b[(j - 1) × pd b + i - 1].

If order = Nag_RowMajor, Bᵢⱼ is stored in b[(i - 1) × pd b + j - 1].

If uplo = Nag_Upper and trans = Nag_NoTrans or if uplo = Nag_Lower and trans = Nag_Trans or trans = Nag_ConjTrans, B is upper triangular and the elements of the array below the diagonal are not set.

If uplo = Nag_Lower and trans = Nag_NoTrans or if uplo = Nag_Upper and trans = Nag_Trans or trans = Nag_ConjTrans, B is lower triangular and the elements of the array above the diagonal are not set.
9: **pdb** – Integer
   
   *Input*

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

   *Constraint*: $pdb \geq \max(1, n)$.

10: **fail** – NagError *

   *Input/Output*

   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 *value* had an illegal value.

**NE_INT**

On entry, $n = \langle value \rangle$.

*Constraint*: $n \geq 0$.

**NE_INT_2**

On entry, $pdb = \langle value \rangle$, $n = \langle value \rangle$.

*Constraint*: $pdb \geq \max(1, n)$.

On entry, $pdb = \langle value \rangle$, $n = \langle value \rangle$.

*Constraint*: $pdb \geq \max(1, n)$.

**NE_INTERNAL_ERROR**

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**

Not applicable.

9 **Further Comments**

None.

10 **Example**

Initializes a 4 by 4 lower triangular matrix $A$ and copies its conjugate transpose to the upper triangular part of $B$. 
10.1 Program Text

/* nag_ztr_copy (f16tec) Example Program.
 * Copyright 2014 Numerical Algorithms Group.
 * Mark 8, 2005.
 */

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

int main(void)
{
    /* Scalars */
    Complex alpha, diag;
    Integer exit_status, n, pda, pdb;
    /* Arrays */
    Complex *a = 0, *b = 0;
    char nag_enum_arg[40];
    /* Nag Types */
    NagError fail;
    Nag_OrderType order;
    Nag_UploType uplo;
    Nag_MatrixType matrix;
    #ifdef NAG_COLUMN_MAJOR
    order = Nag_ColMajor;
    #else
    order = Nag_RowMajor;
    #endif
    exit_status = 0;
    INIT_FAIL(fail);
    printf("nag_ztr_copy (f16tec) Example Program Results\n\n");
    /* Skip heading in data file */
    #ifdef _WIN32
    scanf_s("%*[\n]");
    #else
    scanf("%*[\n]");
    #endif
    /* Read the problem dimension */
    #ifdef _WIN32
    scanf_s("%"NAG_IFMT"%*[\n]", &n);
    #else
    scanf("%"NAG_IFMT"%*[\n]", &n);
    #endif
    /* Read the uplo parameter */
    #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);
    /* Read scalar parameters */
    #ifdef _WIN32
    scanf_s(" ( %lf , %lf ) ( %lf , %lf )%*[\n]",
    "

f16tec.4

NAG Library Manual

Mark 25
#else
    scanf("( %lf, %lf ) ( %lf, %lf )", &alpha.re, &alpha.im, &diag.re, &diag.im);
#endif

pda = n;
pdb = n;

if (n > 0)
{
    /* Allocate memory */
    if (!(a = NAG_ALLOC(n*n, Complex)) ||
        !(b = NAG_ALLOC(n*n, Complex)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
}
else
{
    printf("Invalid n\n");
    exit_status = 1;
    return exit_status;
}

/* nag_ztr_load (f16tgc).
 * Initialize complex triangular matrix.
 *
*/
   nag_ztr_load(order, uplo, n, alpha, diag, a, pda, &fail);
   if (fail.code != NE_NOERROR)
   {
       printf("Error from nag_ztr_laod.\n%s\n", fail.message);
       exit_status = 1;
       goto END;
   }

/* nag_ztr_copy (f16tec).
 * Copies a complex triangular matrix.
 *
*/
   nag_ztr_copy(order, uplo, Nag_ConjTrans, Nag_NonUnitDiag, n, a, pda, b, pdb,
                &fail);
   if (fail.code != NE_NOERROR)
   {
       printf("Error from nag_ztr_copy (f16tec).\n%s\n", fail.message);
       exit_status = 1;
       goto END;
   }

if (uplo == Nag_Upper)
{
    matrix = Nag_LowerMatrix;
}
else
{
    matrix = Nag_UpperMatrix;
}

/* Print generated matrix A */
/* nag_gen_complx_mat_print_comp (x04dbc).
 * Print complex general matrix (comprehensive)
 */
   fflush(stdout);
   nag_gen_complx_mat_print_comp(order, matrix, Nag_NonUnitDiag, n, n, b, pdb,
                                 NagBracketForm, "%5.2f", "Copied Matrix B",
                                 Nag_IntegerLabels, 0, Nag_IntegerLabels, 0, 80,
                                 0, 0, &fail);

   if (fail.code != NE_NOERROR)
10.2 Program Data

\textbf{nag\_ztr\_copy (f16tec)} Example Program Data

\begin{verbatim}
4 : n the dimension of matrix A  
Nag\_Lower : uplo  
( 0.5,-0.3) ( 9.0, 0.0) : alpha, diag
\end{verbatim}

10.3 Program Results

\textbf{nag\_ztr\_copy (f16tec)} Example Program Results

\begin{verbatim}
Copied Matrix B
\begin{tabular}{cccc}
1 & ( 9.00,-0.00) & ( 0.50, 0.30) & ( 0.50, 0.30) & ( 0.50, 0.30) \\
2 & ( 9.00,-0.00) & ( 0.50, 0.30) & ( 0.50, 0.30) & \\
3 & ( 9.00,-0.00) & ( 0.50, 0.30) & ( 0.50, 0.30) & \\
4 & ( 9.00,-0.00) & & &
\end{tabular}
\end{verbatim}