NAG C Library Function Document

nag_ztrexc (f08qtc)

1 Purpose
nag_ztrexc (f08qtc) reorders the Schur factorization of a complex general matrix.

2 Specification

```c
void nag_ztrexc (Nag_OrderType order, Nag_ComputeQType compq,
                 Integer n, Complex t[], Integer pdt, Complex q[], Integer pdq,
                 Integer ifst, Integer ilst, NagError *fail)
```

3 Description
nag_ztrexc (f08qtc) reorders the Schur factorization of a complex general matrix $A = QTQ^H$, so that the diagonal element of $T$ with row index ifst is moved to row ilst.

The reordered Schur form $\tilde{T}$ is computed by a unitary similarity transformation: $\tilde{T} = Z^HTZ$. Optionally the updated matrix $\tilde{Q}$ of Schur vectors is computed as $\tilde{Q} = QZ$, giving $A = \tilde{Q}\tilde{T}\tilde{Q}^H$.

4 References

5 Parameters

1: `order` – Nag_OrderType
   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: `compq` – Nag_ComputeQType
   On entry: indicates whether the matrix $Q$ of Schur vectors is to be updated, as follows:
   - If `compq = Nag_UpdateSchur`, the matrix $Q$ of Schur vectors is updated;
   - If `compq = Nag_NotQ`, no Schur vectors are updated.
   Constraint: `compq = Nag_UpdateSchur` or `Nag_NotQ`.

3: `n` – Integer
   On entry: $n$, the order of the matrix $T$.
   Constraint: $n \geq 0$.

4: `t[dim]` – Complex
   Input/Output
   Note: the dimension, `dim`, of the array `t` must be at least $\max(1, \text{pdt} \times n)$.

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

   On entry: the $n$ by $n$ upper triangular matrix $T$, as returned by nag_zhseqr (f08psc).
On exit: $t$ is overwritten by the updated matrix $\tilde{T}$.

5: **pdt** – Integer  
   
   **Input**
   
   *On entry:* the stride separating matrix row or column elements (depending on the value of order) in the array $t$.
   
   **Constraint:** $pdt \geq \max(1, n)$.

6: **q**[**dim**] – Complex  
   
   **Input/Output**
   
   **Note:** the dimension, **dim**, of the array **q** must be at least
   
   $\max(1, pdq \times n)$ when **compq** = **Nag_UpdateSchur**;
   
   1 when **compq** = **Nag_NotQ**.
   
   If order = **Nag_ColMajor**, the $(i,j)$th element of the matrix $Q$ is stored in $q[(j-1) \times pdq + i - 1]$ and if order = **Nag_RowMajor**, the $(i,j)$th element of the matrix $Q$ is stored in $q[(i-1) \times pdq + j - 1]$.
   
   *On entry:* if **compq** = **Nag_UpdateSchur**, **q** must contain the $n$ by $n$ unitary matrix $Q$ of Schur vectors.
   
   *On exit:* if **compq** = **Nag_UpdateSchur**, **q** contains the updated matrix of Schur vectors.
   
   **q** is not referenced if **compq** = **Nag_NotQ**.

7: **pdq** – Integer  
   
   **Input**
   
   *On entry:* the stride separating matrix row or column elements (depending on the value of order) in the array **q**.
   
   **Constraints:**
   
   if **compq** = **Nag_UpdateSchur**, $pdq \geq \max(1, n)$;
   
   if **compq** = **Nag_NotQ**, $pdq \geq 1$.

8: **ifst** – Integer  
   
   **Input**

9: **ilst** – Integer  
   
   **Input**

   *On entry:* **ifst** and **ilst** must specify the reordering of the diagonal elements of $T$. The element with row index **ifst** is moved to row **ilst** by a sequence of exchanges between adjacent elements.
   
   **Constraint:** $1 \leq \text{ifst} \leq n$ and $1 \leq \text{ilst} \leq n$.

10: **fail** – NagError *  
    
    **Output**

   The NAG error parameter (see the Essential Introduction).

### 6 Error Indicators and Warnings

**NE_INT**

On entry, $n = \langle \text{value} \rangle$.

Constraint: $n \geq 0$.

On entry, $pdt = \langle \text{value} \rangle$.

Constraint: $pdt > 0$.

On entry, $pdq = \langle \text{value} \rangle$.

Constraint: $pdq > 0$.

**NE_INT_2**

On entry, $pdt = \langle \text{value} \rangle$, $n = \langle \text{value} \rangle$.

Constraint: $pdt \geq \max(1, n)$.
On entry, \( n = \langle \text{value} \rangle, \) \( \text{ifst} = \langle \text{value} \rangle, \) \( \text{ilst} = \langle \text{value} \rangle. \)
Constraint: \( 1 \leq \text{ifst} \leq n \) and \( 1 \leq \text{ilst} \leq n. \)

On entry, \( \text{compq} = \langle \text{value} \rangle, \) \( n = \langle \text{value} \rangle, \) \( \text{pdq} = \langle \text{value} \rangle. \)
Constraint: if \( \text{compq} = \text{Nag UpdateSchur}, \) \( \text{pdq} \geq \max(1, n); \)
if \( \text{compq} = \text{Nag NotQ}, \) \( \text{pdq} \geq 1. \)

Memory allocation failed.

On entry, parameter \( \langle \text{value} \rangle \) had an illegal value.

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please consult NAG for assistance.

The computed matrix \( \tilde{T} \) is exactly similar to a matrix \( T + E, \) where
\[
\|E\|_2 = O(\epsilon)\|T\|_2,
\]
and \( \epsilon \) is the \textit{machine precision}.
The values of the eigenvalues are never changed by the re-ordering.

The total number of real floating-point operations is approximately \( 20nr \) if \( \text{compq} = \text{Nag NotQ}, \) and \( 40nr \) if \( \text{compq} = \text{Nag UpdateSchur}, \) where \( r = |\text{ifst} - \text{ilst}|. \)
The real analogue of this function is \text{nag_dtrexc} (f08qfc).

To reorder the Schur factorization of the matrix \( T \) so that element \( t_{11} \) is moved to \( t_{44}, \) where
\[
T = \begin{pmatrix}
-6.00 - 7.00i & 0.36 - 0.36i & -0.19 + 0.48i & 0.88 - 0.25i \\
0.00 + 0.00i & -5.00 + 2.00i & -0.03 - 0.72i & -0.23 + 0.13i \\
0.00 + 0.00i & 0.00 + 0.00i & 8.00 - 1.00i & 0.94 + 0.53i \\
0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i & 3.00 - 4.00i
\end{pmatrix},
\]

The program text is as follows:

```c
/* nag_ztrexc (f08qtc) Example Program. *
 * Copyright 2001 Numerical Algorithms Group.
 * * Mark 7, 2001. */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagx04.h>

9.1 Program Text

/\textcopyright\ 2001\n*/
int main(void)
{
    /* Scalars */
    Integer i, ifst, ilst, j, n, pdq, pdt;
    Integer exit_status=0;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    Complex *q=0, *t=0;
    #ifdef NAG_COLUMN_MAJOR
    #define T(I,J) t[(J-1)*pdt + I - 1]
    order = Nag_ColMajor;
    #else
    #define T(I,J) t[(I-1)*pdq + J - 1]
    order = Nag_RowMajor;
    #endif

    INIT_FAIL(fail);
    Vprintf("f08qtc Example Program Results\n\n");

    /* Skip heading in data file */
    Vscanf("%*[\n ]");
    Vscanf("%ld%*[\n ]", &n);
    #ifdef NAG_COLUMN_MAJOR
    pdq = 1;
    pdt = n;
    #else
    pdq = 1;
    pdt = n;
    #endif

    /* Allocate memory */
    if ( !(q = NAG_ALLOC(1 * 1, Complex)) ||
         !(t = NAG_ALLOC(n * n, Complex)) )
    {
        Vprintf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    /* Read T from data file */
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= n; ++j)
            Vscanf(" ( %lf , %lf )", &T(i,j).re, &T(i,j).im);
    }
    Vscanf("%*[\n ]");
    Vscanf("%ld%ld%*[\n ]", &ifst, &ilst);

    /* Reorder the Schur factorization T */
    f08qtc(order, Nag_NotQ, n, pdt, q, pdq, ifst, ilst, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from f08qtc.\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

    /* Print reordered Schur form */
    x04dbc(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, n,
           t, pdt, Nag_BracketForm, "%7.4f",
           "Reordered Schur form", Nag_IntegerLabels,
           0, Nag_IntegerLabels, 0, 80, 0, 0, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from x04dbc.\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

    END:
    if (q) NAG_FREE(q);
}
if (t) NAG_FREE(t);
return exit_status;
}

9.2 Program Data

f08qtc Example Program Data

<table>
<thead>
<tr>
<th>Value of N</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-6.00, -7.00)</td>
</tr>
<tr>
<td>(0.00, 0.00)</td>
</tr>
<tr>
<td>(0.00, 0.00)</td>
</tr>
<tr>
<td>(0.00, 0.00)</td>
</tr>
</tbody>
</table>

:End of matrix T

<table>
<thead>
<tr>
<th>Values of IFST and ILST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

9.3 Program Results

f08qtc Example Program Results

Reordered Schur form

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-5.0000, 2.0000)</td>
<td>(-0.1574, 0.7143)</td>
<td>(0.1781, -0.1913)</td>
<td>(0.3950, 0.3861)</td>
</tr>
<tr>
<td>(0.0000, 0.0000)</td>
<td>(8.0000, -1.0000)</td>
<td>(1.0742, 0.1447)</td>
<td>(0.2515, -0.3397)</td>
</tr>
<tr>
<td>(0.0000, 0.0000)</td>
<td>(0.0000, 0.0000)</td>
<td>(3.0000, -4.0000)</td>
<td>(0.2264, 0.8962)</td>
</tr>
<tr>
<td>(0.0000, 0.0000)</td>
<td>(0.0000, 0.0000)</td>
<td>(0.0000, 0.0000)</td>
<td>(-6.0000, -7.0000)</td>
</tr>
</tbody>
</table>