NAG C Library Function Document

nag_ztrexc (f08qtc)

1 Purpose

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

2 Specification

```
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: $\tilde{T} = Z^H T Z$. 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

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

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: **compq** – Nag_ComputeQType *Input*

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 *Input*

On entry: n, the order of the matrix T.

Constraint: **n** ≥ 0.

4: **t**[dim] – Complex *Input/Output*

Note: the dimension, dim, of the array **t** must be at least $\max(1, \textbf{pdt} \times \textbf{n})$.

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 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** $\geq \max(1, \mathbf{n})$.

6: **q**[dim] – Complex *Input/Output*

Note: the dimension, dim, of the array **q** must be at least
$\max(1, \mathbf{pdq} \times \mathbf{n})$ 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**$[(j-1) \times \mathbf{pdq} + i - 1]$ and if **order** = **Nag_RowMajor**, the (i, j)th element of the matrix Q is stored in **q**$[(i-1) \times \mathbf{pdq} + j - 1]$.

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** $\geq \max(1, \mathbf{n})$;
if **compq** = **Nag_NotQ**, **pdq** ≥ 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: $1 \leq \mathbf{ifst} \leq \mathbf{n}$ and $1 \leq \mathbf{ilst} \leq \mathbf{n}$.

10: **fail** – NagError * *Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

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

On entry, **pdt** = ⟨*value*⟩.
Constraint: **pdt** > 0.

On entry, **pdq** = ⟨*value*⟩.
Constraint: **pdq** > 0.

NE_INT_2

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

NE_INT_3

On entry, **n** = ⟨*value*⟩, **ifst** = ⟨*value*⟩, **ilst** = ⟨*value*⟩.
Constraint: $1 \leq$ **ifst** $\leq$ **n** and $1 \leq$ **ilst** $\leq$ **n**.

NE_ENUM_INT_2

On entry, **compq** = ⟨*value*⟩, **n** = ⟨*value*⟩, **pdq** = ⟨*value*⟩.
Constraint: if **compq** = **Nag_UpdateSchur**, **pdq** $\geq \max(1, \mathbf{n})$;
if **compq** = **Nag_NotQ**, **pdq** ≥ 1.

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

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

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 consult NAG for assistance.

7 Accuracy

The computed matrix $\tilde{T}$ is exactly similar to a matrix $T + E$, where

$$\|E\|_2 = O(\epsilon)\|T\|_2,$$

and ϵ is the ***machine precision***.

The values of the eigenvalues are never changed by the re-ordering.

8 Further Comments

The total number of real floating-point operations is approximately $20nr$ if **compq** = **Nag_NotQ**, and $40nr$ if **compq** = **Nag_UpdateSchur**, where $r = |$**ifst** $-$ **ilst**$|$.

The real analogue of this function is nag_dtrexc (f08qfc).

9 Example

To reorder the Schur factorization of the matrix T so that element t_{11} is moved to t_{44}, where

$$T = \begin{pmatrix} -6.00 - 7.00i & 0.36 - 0.36i & -0.19 + 0.48i & 0.88 - 0.25i \\ 0.00 + 0.00i & -5.00 + 2.00i & -0.03 - 0.72i & -0.23 + 0.13i \\ 0.00 + 0.00i & 0.00 + 0.00i & 8.00 - 1.00i & 0.94 + 0.53i \\ 0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i & 3.00 - 4.00i \end{pmatrix}.$$

9.1 Program Text

```
/* 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>
```

```
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)*pdt + 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, t, 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
  4                                                          :Value of N
 (-6.00,-7.00) ( 0.36,-0.36) (-0.19, 0.48) ( 0.88,-0.25)
 ( 0.00, 0.00) (-5.00, 2.00) (-0.03,-0.72) (-0.23, 0.13)
 ( 0.00, 0.00) ( 0.00, 0.00) ( 8.00,-1.00) ( 0.94, 0.53)
 ( 0.00, 0.00) ( 0.00, 0.00) ( 0.00, 0.00) ( 3.00,-4.00)   :End of matrix T
  1  4                                        :Values of IFST and ILST
```

9.3 Program Results

```
f08qtc Example Program Results

 Reordered Schur form
                    1                   2                   3                   4
 1  (-5.0000, 2.0000)  (-0.1574, 0.7143)  ( 0.1781,-0.1913)  ( 0.3950, 0.3861)
 2  ( 0.0000, 0.0000)  ( 8.0000,-1.0000)  ( 1.0742, 0.1447)  ( 0.2515,-0.3397)
 3  ( 0.0000, 0.0000)  ( 0.0000, 0.0000)  ( 3.0000,-4.0000)  ( 0.2264, 0.8962)
 4  ( 0.0000, 0.0000)  ( 0.0000, 0.0000)  ( 0.0000, 0.0000)  (-6.0000,-7.0000)
```