NAG Library Function Document

nag_ztrexc (f08qtc)

1 Purpose

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

2 Specification

#include <nag.h>
#include <nagf08.h>

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 \( \text{ifst} \) is moved to row \( \text{ilst} \).

The reordered Schur form \( \tilde{T} \) is computed by a unitary similarity transformation: \( \tilde{T} = Z^H TZ \). 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 Arguments

1: \( \text{order} \) – Nag_OrderType

\textit{Input}

On entry: the \( \text{order} \) 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 \( \text{order} = \text{Nag_RowMajor} \). See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

Constraint: \( \text{order} = \text{Nag_RowMajor} \) or \( \text{Nag_ColMajor} \).

2: \( \text{compq} \) – Nag_ComputeQType

\textit{Input}

On entry: indicates whether the matrix \( Q \) of Schur vectors is to be updated.

\( \text{compq} = \text{Nag_UpdateSchur} \)

The matrix \( Q \) of Schur vectors is updated.

\( \text{compq} = \text{Nag_NotQ} \)

No Schur vectors are updated.

Constraint: \( \text{compq} = \text{Nag_UpdateSchur} \) or \( \text{Nag_NotQ} \).

3: \( n \) – Integer

\textit{Input}

On entry: \( n \), the order of the matrix \( T \).

Constraint: \( n \geq 0 \).
4: \( \text{t}[\text{dim}] \) – Complex 

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

The \((i,j)\)th element of the matrix \( \text{T} \) is stored in

\[
\text{t}[(j-1) \times \text{pd}\text{t} + i - 1] \quad \text{when order} = \text{Nag\_ColMajor};
\]

\[
\text{t}[(i-1) \times \text{pd}\text{t} + j - 1] \quad \text{when order} = \text{Nag\_RowMajor}.
\]

**On entry:** the \( n \) by \( n \) upper triangular matrix \( \text{T} \), as returned by nag_zhseqr (f08psc).

**On exit:** \( \text{t} \) is overwritten by the updated matrix \( \tilde{\text{T}} \).

5: \( \text{pd}\text{t} \) – Integer 

**Input**

**On entry:** the stride separating row or column elements (depending on the value of \( \text{order} \)) in the array \( \text{t} \).

**Constraint:** \( \text{pd}\text{t} \geq \max(1, n) \).

6: \( \text{q}[\text{dim}] \) – Complex 

**Input/Output**

**Note:** the dimension, \( \text{dim} \), of the array \( \text{q} \) must be at least

\[
\max(1, \text{pd}\text{q} \times n) \quad \text{when compq} = \text{Nag\_UpdateSchur};
\]

\[
1 \quad \text{when compq} = \text{Nag\_NotQ}.
\]

The \((i,j)\)th element of the matrix \( \text{Q} \) is stored in

\[
\text{q}[(j-1) \times \text{pd}\text{q} + i - 1] \quad \text{when order} = \text{Nag\_ColMajor};
\]

\[
\text{q}[(i-1) \times \text{pd}\text{q} + j - 1] \quad \text{when order} = \text{Nag\_RowMajor}.
\]

**On entry:** if \( \text{compq} = \text{Nag\_UpdateSchur} \), \( \text{q} \) must contain the \( n \) by \( n \) unitary matrix \( \text{Q} \) of Schur vectors.

**On exit:** if \( \text{compq} = \text{Nag\_UpdateSchur} \), \( \text{q} \) contains the updated matrix of Schur vectors.

If \( \text{compq} = \text{Nag\_NotQ} \), \( \text{q} \) is not referenced.

7: \( \text{pd}\text{q} \) – Integer 

**Input**

**On entry:** the stride separating row or column elements (depending on the value of \( \text{order} \)) in the array \( \text{q} \).

**Constraints:**

\[
\text{if compq} = \text{Nag\_UpdateSchur} \quad \text{pd}\text{q} \geq \max(1, n);
\]

\[
\text{if compq} = \text{Nag\_NotQ} \quad \text{pd}\text{q} \geq 1.
\]

8: \( \text{ifst} \) – Integer 

**Input**

9: \( \text{ilst} \) – Integer 

**Input**

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

**Constraint:** \( 1 \leq \text{ifst} \leq n \) and \( 1 \leq \text{ilst} \leq n \).

10: \( \text{fail} \) – 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_ENUM_INT_2
On entry, compq = <value>, pdq = <value> and n = <value>.
Constraint: if compq = Nag_UpdateSchur, pdq ≥ max(1, n);
if compq = Nag_NotQ, pdq ≥ 1.

NE_INT
On entry, n = <value>.
Constraint: n ≥ 0.
On entry, pdq = <value>.
Constraint: pdq > 0.
On entry, pdt = <value>.
Constraint: pdt > 0.

NE_INT_2
On entry, pdt = <value> and n = <value>.
Constraint: pdt ≥ max(1, n).

NE_INT_3
On entry, n = <value>, ifst = <value> and ilst = <value>.
Constraint: 1 ≤ ifst ≤ n and 1 ≤ ilst ≤ n.

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

8 Parallelism and Performance
nag_ztrexc (f08qtc) is not threaded by NAG in any implementation.
nag_ztrexc (f08qtc) 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.

Mark 25
9  Further Comments

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

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

10  Example

This example reorders the Schur factorization of the matrix $T$ so that element $t_{11}$ is moved to $t_{44}$, where

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

10.1  Program Text

/* nag_ztrexc (f08qtc) Example Program. */
/* Copyright 2014 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_LOAD_FP
    /* The following line is needed to force the Microsoft linker
to load floating point support */
    float force_loading_of_ms_float_support = 0;
    #endif /* NAG_LOAD_FP */

    #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);

    printf("nag_ztrexc (f08qtc) Example Program Results\n\n");

    /* Skip heading in data file */
    #ifdef _WIN32
    scanf_s("%*[\n] ");
    #else
    scanf("%*[\n] ");
    #endif
    #ifdef _WIN32
    scanf_s("%"NAG_IFMT"%*[\n] ", &n);
    #else
    scanf("%"NAG_IFMT"%*[\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)))
{
    printf("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)
    {
        #ifdef _WIN32
            scanf_s(" ( %lf , %lf )", &T(i, j).re, &T(i, j).im);
        #else
            scanf(" ( %lf , %lf )", &T(i, j).re, &T(i, j).im);
        #endif
    }
    #ifdef _WIN32
        scanf_s("%*[\n]");
    #else
        scanf("%*[\n]");
    #endif
    #ifdef _WIN32
        scanf_s("%NAG_IFMT%NAG_IFMT%*[\n] ", &ifst, &ilst);
    #else
        scanf("%NAG_IFMT%NAG_IFMT%*[\n] ", &ifst, &ilst);
    #endif

/* Reorder the Schur factorization T */
    /\  nag_ztrexc (f08qtc).
    /\  Reorder Schur factorization of complex matrix using
    /\  unitary similarity transformation
    /\  *\n
    nag_ztrexc(order, Nag_NotQ, n, t, pdt, q, pdq, ifst, ilst, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_ztrexc (f08qtc).\n%\s\n", fail.message);
        exit_status = 1;
        goto END;
    }

    /\  Print reordered Schur form */
    /\  nag_gen_complx_mat_print_comp (x04dbc).
    /\  Print complex general matrix (comprehensive)
    /\  *\n
    fflush(stdout);
    nag_gen_complx_mat_print_comp(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)
    {
        printf("Error from nag_gen_complx_mat_print_comp (x04dbc).\n%\s\n", fail.message);
        exit_status = 1;
        goto END;
    }

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

10.2 Program Data

nag_ztrexc (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

10.3 Program Results

nag_ztrexc (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)