NAG Library Function Document

nag_ztrsen (f08quc)

1 Purpose

nag_ztrsen (f08quc) reorders the Schur factorization of a complex general matrix so that a selected cluster of eigenvalues appears in the leading elements on the diagonal of the Schur form. The function also optionally computes the reciprocal condition numbers of the cluster of eigenvalues and/or the invariant subspace.

2 Specification

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

void nag_ztrsen (Nag_OrderType order, Nag_JobType job,
                 Nag_ComputeQType compq, const Nag_Boolean select[],
                 Integer n, Complex t[], Integer pdt, Complex q[], Integer pdq,
                 Complex w[], Integer *m, double *s, double *sep, NagError *fail)
```

3 Description

nag_ztrsen (f08quc) reorders the Schur factorization of a complex general matrix $A = QTQ^H$, so that a selected cluster of eigenvalues appears in the leading diagonal elements of the Schur form.

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 $A = \tilde{Q}\tilde{T}\tilde{Q}^H$.

Let $\tilde{T} = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix}$, where the selected eigenvalues are precisely the eigenvalues of the leading $m$ by $m$ sub-matrix $T_{11}$. Let $\tilde{Q}$ be correspondingly partitioned as $(Q_1 \ Q_2)$ where $Q_1$ consists of the first $m$ columns of $Q$. Then $AQ_1 = Q_1T_{11}$, and so the $m$ columns of $Q_1$ form an orthonormal basis for the invariant subspace corresponding to the selected cluster of eigenvalues.

Optionally the function also computes estimates of the reciprocal condition numbers of the average of the cluster of eigenvalues and of the invariant subspace.

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:  
   **job** – Nag_JobType  
   *Input*

   *On entry*: indicates whether condition numbers are required for the cluster of eigenvalues and/or
   the invariant subspace.

   - **job** = **Nag_DoNothing**
     No condition numbers are required.
   - **job** = **Nag_EigVals**
     Only the condition number for the cluster of eigenvalues is computed.
   - **job** = **Nag_Subspace**
     Only the condition number for the invariant subspace is computed.
   - **job** = **Nag_DoBoth**
     Condition numbers for both the cluster of eigenvalues and the invariant subspace are
     computed.

   *Constraint*: **job** = **Nag_DoNothing**, **Nag_EigVals**, **Nag_Subspace** or **Nag_DoBoth**.

3:  
   **compq** – Nag_ComputeQType  
   *Input*

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

   - **compq** = **Nag_UpdateSchur**
     The matrix \( Q \) of Schur vectors is updated.
   - **compq** = **Nag_NotQ**
     No Schur vectors are updated.

   *Constraint*: **compq** = **Nag_UpdateSchur** or **Nag_NotQ**.

4:  
   **select**[**dim**] – const Nag_Boolean  
   *Input*

   *Note*: the dimension, **dim**, of the array **select** must be at least \( \max(1, n) \).

   *On entry*: specifies the eigenvalues in the selected cluster. To select a complex eigenvalue \( \lambda_j \),
   **select**[**j** − 1] must be set **Nag_TRUE**.

5:  
   **n** – Integer  
   *Input*

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

   *Constraint*: \( n \geq 0 \).

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

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

   The \((i, j)\)th element of the matrix \( T \) is stored in
   
   \[
   t[(j - 1) \times pdt + i - 1] \quad \text{when } order = \text{Nag_ColMajor;}
   \]
   
   
   \[
   t[(i - 1) \times pdt + j - 1] \quad \text{when } order = \text{Nag_RowMajor.}
   \]

   *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 \( \tilde{T} \).

7:  
   **pdt** – Integer  
   *Input*

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

   *Constraint*: \( pdt \geq \max(1, n) \).
8: \( q[dim] \) – Complex

**Note:** the dimension, \( dim \), of the array \( q \) must be at least
\[
\max(1, pdq \times n) \quad \text{when} \quad \text{compq} = \text{Nag}._{\text{UpdateSchur}};
\]
\[
1 \quad \text{when} \quad \text{compq} = \text{Nag}._{\text{NotQ}}.
\]
The \((i, j)\)th element of the matrix \( Q \) is stored in
\[
q[(j - 1) \times pdq + i - 1] \quad \text{when} \quad \text{order} = \text{Nag}._{\text{ColMajor}};
\]
\[
q[(i - 1) \times pdq + j - 1] \quad \text{when} \quad \text{order} = \text{Nag}._{\text{RowMajor}}.
\]

**On entry:** if \( \text{compq} = \text{Nag}._{\text{UpdateSchur}}, q \) must contain the \( n \) by \( n \) unitary matrix \( Q \) of Schur vectors, as returned by \text{nag_zhseqr} (f08psc).

**On exit:** if \( \text{compq} = \text{Nag}._{\text{UpdateSchur}}, q \) contains the updated matrix of Schur vectors; the first \( m \) columns of \( Q \) form an orthonormal basis for the specified invariant subspace.

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

9: \( pdq \) – Integer

**Input**

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

**Constraints:**
\[
\begin{align*}
&\text{if} \quad \text{compq} = \text{Nag}._{\text{UpdateSchur}}, \quad pdq \geq \max(1, n); \\
&\text{if} \quad \text{compq} = \text{Nag}._{\text{NotQ}}, \quad pdq \geq 1.
\end{align*}
\]

10: \( w[dim] \) – Complex

**Output**

**Note:** the dimension, \( dim \), of the array \( w \) must be at least \( \max(1, n) \).

**On exit:** the reordered eigenvalues of \( \tilde{T} \). The eigenvalues are stored in the same order as on the diagonal of \( \tilde{T} \).

11: \( m \) – Integer *

**Output**

**On exit:** \( m \), the dimension of the specified invariant subspace, which is the same as the number of selected eigenvalues (see \( \text{select} \)); \( 0 \leq m \leq n \).

12: \( s \) – double *

**Output**

**On exit:** if \( \text{job} = \text{Nag}._{\text{EigVals}} \) or \( \text{Nag}._{\text{DoBoth}}, s \) is a lower bound on the reciprocal condition number of the average of the selected cluster of eigenvalues. If \( m = 0 \) or \( n \), \( s = 1 \).

If \( \text{job} = \text{Nag}._{\text{DoNothing}} \) or \( \text{Nag}._{\text{Subspace}}, s \) is not referenced.

13: \( sep \) – double *

**Output**

**On exit:** if \( \text{job} = \text{Nag}._{\text{Subspace}} \) or \( \text{Nag}._{\text{DoBoth}}, sep \) is the estimated reciprocal condition number of the specified invariant subspace. If \( m = 0 \) or \( n \), \( sep = ||T|| \).

If \( \text{job} = \text{Nag}._{\text{DoNothing}} \) or \( \text{Nag}._{\text{EigVals}}, sep \) is not referenced.

14: \( \text{fail} \) – 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_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 similar to a matrix $(T + E)$, where

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

and $\epsilon$ is the *machine precision*.

$s$ cannot underestimate the true reciprocal condition number by more than a factor of $\sqrt{\min(m, n-m)}$.

$sep$ may differ from the true value by $\sqrt{m(n-m)}$. The angle between the computed invariant subspace and the true subspace is

$$
\frac{O(\epsilon)\|A\|_2}{sep}.
$$

The values of the eigenvalues are never changed by the reordering.

8 **Parallelism and Performance**

nag_ztrsne (f08quc) is not threaded by NAG in any implementation.

nag_ztrsne (f08quc) 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.
Further Comments

The real analogue of this function is nag_dtrsen (f08qgc).

Example

This example reorders the Schur factorization of the matrix $A = QTQ^H$ such that the eigenvalues stored in elements $t_{11}$ and $t_{44}$ appear as the leading elements on the diagonal of the reordered matrix $\tilde{T}$, where

$$T = \begin{pmatrix}
-6.0004 - 6.9999i & 0.3637 - 0.3656i & -0.1880 + 0.4787i & 0.8785 - 0.2539i \\
0.0000 + 0.0000i & -5.0000 + 2.0060i & -0.0307 - 0.7217i & -0.2290 + 0.1313i \\
0.0000 + 0.0000i & 0.0000 + 0.0000i & 7.9982 - 0.9964i & 0.9357 + 0.5359i \\
0.0000 + 0.0000i & 0.0000 + 0.0000i & 0.0000 + 0.0000i & 3.0023 - 3.9998i
\end{pmatrix}$$

and

$$Q = \begin{pmatrix}
-0.8347 - 0.1364i & -0.0628 + 0.3806i & 0.2765 - 0.0846i & 0.0633 - 0.2199i \\
0.0664 - 0.2968i & 0.2365 + 0.5240i & -0.5877 - 0.4208i & 0.0835 + 0.2183i \\
-0.0362 - 0.3215i & 0.3143 - 0.5473i & 0.0576 - 0.5736i & 0.0057 - 0.4058i \\
0.0086 + 0.2958i & -0.3416 - 0.0757i & -0.1900 - 0.1600i & 0.8327 - 0.1868i
\end{pmatrix}.$$  

The original matrix $A$ is given in Section 10 in nag_zunghr (f08ntc).

10.1 Program Text

/* nag_ztrsen (f08quc) Example Program. */
/* Copyright 2014 Numerical Algorithms Group. */
/* Mark 7, 2001. */

#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagx04.h>
#include <nagf16.h>
#include <nagx02.h>

int main(void)
{
    /* Scalars */
    Integer i, j, m, n, pda, pdc, pdq, pdt, select_len, w_len;
    Integer exit_status = 0;
    double norm, s, sep;
    Complex alpha, beta;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    Complex *a = 0, *c = 0, *q = 0, *t = 0, *w = 0;
    char nag_enum_arg[40];
    Nag_Boolean *select = 0;
    #ifdef NAG_COLUMN_MAJOR
    #define T(I, J) t[(J-1)*pdt +I-1 ]
    #define Q(I, J) q[(J-1)*pdq +I-1 ]
    order = Nag_ColMajor;
    #else
    #define T(I, J) t[(I-1)*pdt+J-1 ]
    #define Q(I, J) q[(I-1)*pdq +J-1 ]
    order = Nag_RowMajor;
    #endif
    INIT_FAIL(fail);
printf("nag_ztrsne (f08quc) 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);
#endif
#ifdef _WIN32
    scanf_s("%"NAG_IFMT"%*[\n] ", &n);
#else
    scanf("%"NAG_IFMT"%*[\n] ", &n);
#endif

/* Allocate memory */
if (!(a = NAG_ALLOC(n * n, Complex)) ||
    !(c = NAG_ALLOC(n * n, Complex)) ||
    !(q = NAG_ALLOC(n * n, Complex)) ||
    !(w = NAG_ALLOC(w_len, Complex)) ||
    !(select = NAG_ALLOC(select_len, Nag_Boolean)) ||
    !(t = NAG_ALLOC(n * n, Complex)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Read T and Q 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
    for (i = 0; i < n; ++i)
    {
        for (j = 1; j <= n; ++j)
        {
            #ifdef _WIN32
                scanf_s(" ( %lf , %lf ) ", &Q(i, j).re, &Q(i, j).im);
            #else
                scanf(" ( %lf , %lf ) ", &Q(i, j).re, &Q(i, j).im);
            #endif
        }
        #ifdef _WIN32
            scanf_s("%*[\n] ");
        #else
            scanf("%*[\n] ");
        #endif
    }
}
ifdef _WIN32
    scanf_s("%39s ", nag_enum_arg, _countof(nag_enum_arg));
#else
    scanf("%39s ", nag_enum_arg);
#endif

/* nag_enum_name_to_value (x04nac).
 * Converts NAG enum member name to value
 */
select[i] = (Nag_Boolean) nag_enum_name_to_value(nag_enum_arg);
#endif

if (fail.code != NE_NOERROR)
    printf("Error from nag_zgemm (f16zac).
%s
", fail.message);
    exit_status = 1;
    goto END;

nag_zgemm(order, Nag_NoTrans, Nag_NoTrans, n, n, n, alpha, q, pdq,
t, pdt, beta, c, pdc, &fail);
if (fail.code != NE_NOERROR)
    printf("Error from nag_zgemm (f16zac).
%s
", fail.message);
    exit_status = 1;
    goto END;

nag_zgemm(order, Nag_NoTrans, Nag_ConjTrans, n, n, n, alpha, c, pdc,
q, pdq, beta, a, pda, &fail);
if (fail.code != NE_NOERROR)
    printf("Error from nag_zgemm (f16zac).
%s
", fail.message);
    exit_status = 1;
    goto END;

f08 – Least-squares and Eigenvalue Problems (LAPACK)

f08quc

Mark 25
f08quc.7
/* nag_zgemm (f16zac): Compute A - Qt*Tt*Qt^H from the reordered */
/* Qt and T*/
alpha.re = 1.0;
alp ha.im = 0.0;
b eta.re = 0.0;
b eta.im = 0.0;
nag_zgemm(order, Nag_NoTrans, Nag_NoTrans, n, n, n, alpha, q, pdq,
t, pdt, beta, c, pdc, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zgemm (f16zac).\n%s\n", fail.message);
    exit_status = 1;
goto END;
}
alpha.re = -1.0;
b eta.re = 1.0;
nag_zgemm(order, Nag_NoTrans, Nag_ConjTrans, n, n, n, alpha, c, pdc,
q, pdq, beta, a, pda, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zgemm (f16zac).\n%s\n", fail.message);
    exit_status = 1;
goto END;
}

/* nag_zge_norm (f16uac): Find norm of matrix A and print warning if */
/* it is too large */
nag_zge_norm(order, Nag_OneNorm, n, n, a, pda, &norm, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zge_norm (f16uac).\n%s\n", fail.message);
    exit_status = 1;
goto END;
}
if (norm>pow(x02ajc(),0.8))
{
    printf("%s\n","Norm of A-(Qt*Tt*Qt^H) is much greater than 0.",
    "Schur factorization has failed.");
}
else
{
    /* Print condition number estimates */
    printf("%s\n", "Condition number estimate of the selected cluster of"
    " eigenvalues = %11.2e\n", 1.0/s);
    printf("%s\n", "Condition number estimate of the specified invariant"
    " subspace = %11.2e\n", 1.0/sep);
}

END:
NAG_FREE(a);
NAG_FREE(c);
NAG_FREE(q);
NAG_FREE(t);
NAG_FREE(w);
NAG_FREE(select);
return exit_status;

10.2 Program Data
nag_ztrsen (f08quc) Example Program Data
nag_ztrsen (f08quc) Example Program Data
4 : Value of n
(-6.0004,-6.9999) ( 0.3637,-0.3656) (-0.1880, 0.4787) ( 0.8785,-0.2539)
( 0.0000, 0.0000) (-5.0000, 2.0060) (-0.0307,-0.7217) (-0.2290, 0.1313)
( 0.0000, 0.0000) ( 7.9982,-0.9964) ( 0.9357, 0.5359)
( 0.0000, 0.0000) ( 0.0000, 0.0000) ( 0.0000, 0.0000) ( 3.0023,-3.9998)

f08quc.8   Mark 25
## 10.3 Program Results

**nag_ztrsen (f08quc) Example Program Results**

<table>
<thead>
<tr>
<th>Matrix A</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (-3.9702,-5.0406)</td>
<td>(-4.1108, 3.7002)</td>
<td>(-0.3403, 1.0098)</td>
<td>1.2899,-0.8590</td>
<td></td>
</tr>
<tr>
<td>2 ( 0.3397,-1.5006)</td>
<td>( 1.5201,-0.4301)</td>
<td>( 1.8797,-5.3804)</td>
<td>3.3606, 0.6498</td>
<td></td>
</tr>
<tr>
<td>3 ( 3.3101,-3.8506)</td>
<td>( 2.4996, 3.4504)</td>
<td>( 0.8802,-1.0802)</td>
<td>0.6401,-1.4800</td>
<td></td>
</tr>
<tr>
<td>4 ( -1.0999, 0.8199)</td>
<td>( 1.8103,-1.5905)</td>
<td>( 3.2502, 1.3297)</td>
<td>1.5701,-3.4397</td>
<td></td>
</tr>
</tbody>
</table>

Condition number estimate of the selected cluster of eigenvalues = 1.02e+00

Condition number estimate of the specified invariant subspace = 1.82e-01