NAG C 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

```
void nag_ztrsen (Nag_OrderType order, Nag_JobType job, Nag_ComputeQType compq,
      const 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: $\tilde{T} = Z^H TZ$. Optionally the updated matrix $\tilde{Q}$ of Schur vectors is computed as $\tilde{Q} = QZ$, giving $A = \tilde{Q}\tilde{T}\tilde{Q}^H$.

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 submatrix T_{11}. Let $\tilde{Q}$ be correspondingly partitioned as $(Q_1 \quad Q_2)$ where Q_1 consists of the first m columns of Q. Then $AQ_1 = Q_1 T_{11}$, 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

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: **job** – Nag_JobType *Input*

On entry: indicates whether condition numbers are required for the cluster of eigenvalues and/or the invariant subspace, as follows:

if **job** = **Nag_DoNothing**, no condition numbers are required;

if **job** = **Nag_EigVals**, only the condition number for the cluster of eigenvalues is computed;

if **job** = **Nag_Subspace**, only the condition number for the invariant subspace is computed;

if **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, 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**.

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

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

On entry: specifies the eigenvalues in the selected cluster. To select a complex eigenvalue λ_j, **select**$[j-1]$ must be set **TRUE**.

5: **n** – Integer *Input*

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

Constraint: $\mathbf{n} \geq 0$.

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

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

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

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 $\tilde{T}$.

7: **pdt** – Integer *Input*

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

Constraint: $\mathbf{pdt} \geq \max(1, \mathbf{n})$.

8: **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 $\mathbf{q}[(j-1) \times \mathbf{pdq} + i - 1]$ and if **order** $=$ **Nag_RowMajor**, the (i,j)th element of the matrix Q is stored in $\mathbf{q}[(i-1) \times \mathbf{pdq} + j - 1]$.

On entry: if **compq** $=$ **Nag_UpdateSchur**, **q** must contain the n by n unitary matrix Q of Schur vectors, as returned by nag_zhseqr (f08psc).

On exit: if **compq** $=$ **Nag_UpdateSchur**, **q** contains the updated matrix of Schur vectors; the first m rows or columns of **q** (depending on the value of **order**) form an orthonormal basis for the specified invariant subspace.

q is not referenced if **compq** $=$ **Nag_NotQ**.

9: **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.

10: **w**[dim] – Complex *Output*

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

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 **select**); $0 \leq m \leq n$.

12: **s** – double * *Output*

On exit: if **job** $=$ **Nag_EigVals** or **Nag_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** $= 1$.

s is not referenced if **job** $=$ **Nag_DoNothing** or **Nag_Subspace**.

13: **sep** – double * *Output*

On exit: if **job** $=$ **Nag_Subspace** or **Nag_DoBoth**, **sep** is the estimated reciprocal condition number of the specified invariant subspace. If **m** $= 0$ or **n**, **sep** $= \|T\|$.

sep is not referenced if **job** $=$ **Nag_DoNothing** or **Nag_EigVals**.

14: **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_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 similar to a matrix $T + E$, where

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

and ϵ 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 $\dfrac{O(\epsilon)\|A\|_2}{sep}$.

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

8 Further Comments

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

9 Example

To reorder 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 nag_zunghr (f08ntc).

9.1 Program Text

```
/* nag_ztrsen (f08quc) 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, j, m, n, pdq, pdt, select_len, w_len;
  Integer  exit_status=0;
  double   s, sep;
  NagError fail;
  Nag_OrderType order;
  /* Arrays */
  Complex *q=0, *t=0, *w=0;
  char   sel_char[2];
  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);
  Vprintf("f08quc Example Program Results\n\n");

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

  /* Allocate memory */
  if ( !(q = NAG_ALLOC(n * n, Complex)) ||
       !(w = NAG_ALLOC(w_len, Complex)) ||
       !(select = NAG_ALLOC(select_len, Boolean)) ||
       !(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] ");
  for (i = 1; i <= n; ++i)
    {
      for (j = 1; j <= n; ++j)
        Vscanf(" ( %lf , %lf ) ", &Q(i,j).re, &Q(i,j).im);
    }
  Vscanf("%*[^\n] ");
  for (i = 0; i < n; ++i)
    {
      Vscanf("  %1s ", sel_char);
      if (*(unsigned char *)sel_char == 'F')
        select[i] = FALSE;
      else
        select[i] = TRUE;
    }
  Vscanf("%*[^\n] ");

  /* Reorder the Schur factorization T */
  f08quc(order, Nag_DoBoth, Nag_UpdateSchur, select, n, t, pdt,
         q, pdq, w, &m, &s, &sep, &fail);
  if (fail.code != NE_NOERROR)
```

```
    {
      Vprintf("Error from f08quc.\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;
    }
  /* Print basis of invariant subspace */
  x04dbc(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, m, q, pdq,
         Nag_BracketForm, "%7.4f", "Basis of invariant subspace",
         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;
    }
  /* Print condition number estimates */
  Vprintf("\n Condition number estimate of the selected cluster of"
          " eigenvalues = %10.2e\n",1.0/s);
  Vprintf("\n Condition number estimate of the specified invariant"
          " subspace = %10.2e\n",1.0/sep);
 END:
  if (q) NAG_FREE(q);
  if (t) NAG_FREE(t);
  if (w) NAG_FREE(w);
  if (select) NAG_FREE(select);

  return exit_status;
}
```

9.2 Program Data

```
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) ( 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)
                                                      :End of matrix T
 (-0.8347,-0.1364) (-0.0628, 0.3806) ( 0.2765,-0.0846) ( 0.0633,-0.2199)
 ( 0.0664,-0.2968) ( 0.2365, 0.5240) (-0.5877,-0.4208) ( 0.0835, 0.2183)
 (-0.0362,-0.3215) ( 0.3143,-0.5473) ( 0.0576,-0.5736) ( 0.0057,-0.4058)
 ( 0.0086, 0.2958) (-0.3416,-0.0757) (-0.1900,-0.1600) ( 0.8327,-0.1868)
                                                      :End of matrix Q
  T   F   F   T                                       :End of SELECT
```

9.3 Program Results

```
f08quc Example Program Results

 Reordered Schur form
                  1                  2                  3                  4
 1  (-6.0004,-6.9999)  (-0.9433, 0.0086)  (-0.4839,-0.2426)  ( 0.1539, 0.4000)
 2  ( 0.0000, 0.0000)  ( 3.0023,-3.9998)  ( 0.3028, 0.1519)  ( 1.0421,-0.2338)
 3  ( 0.0000, 0.0000)  ( 0.0000, 0.0000)  (-5.0000, 2.0060)  ( 0.6891,-0.1546)
 4  ( 0.0000, 0.0000)  ( 0.0000, 0.0000)  ( 0.0000, 0.0000)  ( 7.9982,-0.9964)
 Basis of invariant subspace
                  1                  2
 1  ( 0.8458, 0.0000)  ( 0.0488,-0.2073)
 2  (-0.0177, 0.3036)  ( 0.1275, 0.3006)
 3  ( 0.0876, 0.3115)  ( 0.0398,-0.2711)
 4  (-0.0562,-0.2905)  ( 0.8792, 0.0000)

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

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