NAG C Library Function Document

nag_dtrsen (f08qgc)

1 Purpose

nag_dtrsen (f08qgc) reorders the Schur factorization of a real general matrix so that a selected cluster of eigenvalues appears in the leading elements or blocks 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_dtrsen (Nag_OrderType order, Nag_JobType job, Nag_ComputeQType compq,
     const Boolean select[], Integer n, double t[], Integer pdt, double q[],
     Integer pdq, double wr[], double wi[], Integer *m, double *s, double *sep,
     NagError *fail)
```

3 Description

nag_dtrsen (f08qgc) reorders the Schur factorization of a real general matrix $A = QTQ^T$, so that a selected cluster of eigenvalues appears in the leading diagonal elements or blocks of the Schur form.

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

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 $\begin{pmatrix} Q_1 & Q_2 \end{pmatrix}$ 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: the eigenvalues in the selected cluster. To select a real eigenvalue λ_j, **select**[$j-1$] must be set **TRUE**. To select a complex conjugate pair of eigenvalues λ_j and λ_{j+1} (corresponding to a 2 by 2 diagonal block), **select**[$j-1$] and/or **select**[j] must be set to **TRUE**. A complex conjugate pair of eigenvalues **must** be either both included in the cluster or both excluded. See also Section 8.

5: **n** – Integer *Input*

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

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

6: **t**[dim] – double *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 **t**[$(j-1) \times$ **pdt** $+ i - 1$] and if **order** = **Nag_RowMajor**, the (i,j)th element of the matrix T is stored in **t**[$(i-1) \times$ **pdt** $+ j - 1$].

On entry: the n by n upper quasi-triangular matrix T in canonical Schur form, as returned by nag_dhseqr (f08pec). See also Section 8.

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

8: **q**[dim] – double *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**[$(j-1) \times$ **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 orthogonal matrix Q of Schur vectors, as returned by nag_dhseqr (f08pec).

On exit: if **compq** = **Nag_UpdateSchur**, **q** contains the updated matrix of Schur vectors; the first m columns of Q 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: **wr**[dim] – double *Output*
11: **wi**[dim] – double *Output*

Note: the dimensions, dim, of the arrays **wr** and **wi** must each be at least $\max(1, \mathbf{n})$.

On exit: the real and imaginary parts, respectively, of the reordered eigenvalues of $\tilde{T}$. The eigenvalues are stored in the same order as on the diagonal of $\tilde{T}$; see Section 8 for details. Note that if a complex eigenvalue is sufficiently ill-conditioned, then its value may differ significantly from its value before reordering.

12: **m** – Integer * *Output*

On exit: m, the dimension of the specified invariant subspace. The value of m is obtained by counting 1 for each selected real eigenvalue and 2 for each selected complex conjugate pair of eigenvalues (see **select**); $0 \leq m \leq n$.

13: **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; if **fail** = 1 (see Section 6), **s** is set to zero.

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

14: **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** = $\|T\|$; if **fail** = 1 (see Section 6), **sep** is set to zero.

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

15: **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_REORDER

The reordering of T failed because a selected eigenvalue was too close to an eigenvalue which was not selected; this error exit can only occur if at least one of the eigenvalues involved was complex. The problem is too ill-conditioned: consider modifying the selection of eigenvalues so that eigenvalues which are very close together are either all included in the cluster or all excluded. On exit, T may have been partially reordered, but **wr**, **wi** and Q (if requested) are updated consistently with T; **s** and **sep** (if requested) are both set to zero.

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

Note that if a 2 by 2 diagonal block is involved in the re-ordering, its off-diagonal elements are in general changed; the diagonal elements and the eigenvalues of the block are unchanged unless the block is sufficiently ill-conditioned, in which case they may be noticeably altered. It is possible for a 2 by 2 block to break into two 1 by 1 blocks, that is, for a pair of complex eigenvalues to become purely real. The values of real eigenvalues however are never changed by the re-ordering.

8 Further Comments

The input matrix T must be in canonical Schur form, as is the output matrix $\tilde{T}$. This has the following structure.

If all the computed eigenvalues are real, $\tilde{T}$ is upper triangular, and the diagonal elements of $\tilde{T}$ are the eigenvalues; $\mathbf{wr}[i-1] = \tilde{t}_{ii}$ for $i = 1, 2, \ldots, n$ and $\mathbf{wi}[i-1] = 0.0$.

If some of the computed eigenvalues form complex conjugate pairs, then $\tilde{T}$ has 2 by 2 diagonal blocks. Each diagonal block has the form

$$\begin{pmatrix} \tilde{t}_{ii} & \tilde{t}_{i,i+1} \\ \tilde{t}_{i+1,i} & \tilde{t}_{i+1,i+1} \end{pmatrix} = \begin{pmatrix} \alpha & \beta \\ \gamma & \alpha \end{pmatrix}$$

where $\beta\gamma < 0$. The corresponding eigenvalues are $\alpha \pm \sqrt{\beta\gamma}$; $\mathbf{wr}[i-1] = \mathbf{wr}[i] = \alpha$; $\mathbf{wi}[i-1] = +\sqrt{|\beta\gamma|}$; $\mathbf{wi}[i] = -\mathbf{wi}[i-1]$.

The complex analogue of this function is nag_ztrsen (f08quc).

9 Example

To reorder the Schur factorization of the matrix $A = QTQ^T$ such that the two real eigenvalues appear as the leading elements on the diagonal of the reordered matrix $\tilde{T}$, where

$$T = \begin{pmatrix} 0.7995 & -0.1144 & 0.0060 & 0.0336 \\ 0.0000 & -0.0994 & 0.2478 & 0.3474 \\ 0.0000 & -0.6483 & -0.0994 & 0.2026 \\ 0.0000 & 0.0000 & 0.0000 & -0.1007 \end{pmatrix}$$

and

$$Q = \begin{pmatrix} 0.6551 & 0.1037 & 0.3450 & 0.6641 \\ 0.5236 & -0.5807 & -0.6141 & -0.1068 \\ -0.5362 & -0.3073 & -0.2935 & 0.7293 \\ 0.0956 & 0.7467 & -0.6463 & 0.1249 \end{pmatrix}.$$

The original matrix A is given in nag_dorghr (f08nfc).

9.1 Program Text

```
/* nag_dtrsen (f08qgc) 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 */
  double *q=0, *t=0, *wi=0, *wr=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("f08qgc 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, double)) ||
       !(wi = NAG_ALLOC(w_len, double)) ||
       !(wr = NAG_ALLOC(w_len, double)) ||
       !(select = NAG_ALLOC(select_len, Boolean)) ||
       !(t = NAG_ALLOC(n * n, double)) )
    {
      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", &T(i,j));
    }
  Vscanf("%*[^\n] ");
  for (i = 1; i <= n; ++i)
    {
      for (j = 1; j <= n; ++j)
        Vscanf("%lf", &Q(i,j));
    }
  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 */
  f08qgc(order, Nag_DoBoth, Nag_UpdateSchur, select, n, t, pdt,
         q, pdq, wr, wi, &m, &s, &sep, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from f08qgc.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  /* Print reordered Schur form */
  x04cac(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, n,
         t, pdt, "Reordered Schur form", 0, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from x04cac.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  /* Print basis of invariant subspace */
  x04cac(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, m, q, pdq,
         "Basis of invariant subspace", 0, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from x04cac.\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 (wi) NAG_FREE(wi);
  if (wr) NAG_FREE(wr);
  if (select) NAG_FREE(select);

  return exit_status;
}
```

9.2 Program Data

```
f08qgc Example Program Data
  4                                   :Value of N
  0.7995  -0.1144   0.0060   0.0336
  0.0000  -0.0994   0.2478   0.3474
  0.0000  -0.6483  -0.0994   0.2026
  0.0000   0.0000   0.0000  -0.1007   :End of matrix T
  0.6551   0.1037   0.3450   0.6641
  0.5236  -0.5807  -0.6141  -0.1068
 -0.5362  -0.3073  -0.2935   0.7293
  0.0956   0.7467  -0.6463   0.1249   :End of matrix Q
  T   F   F   T                       :End of SELECT
```

9.3 Program Results

```
f08qgc Example Program Results

 Reordered Schur form
          1        2        3        4
 1   0.7995  -0.0059   0.0751  -0.0927
 2  -0.0000  -0.1007  -0.3936   0.3569
 3   0.0000   0.0000  -0.0994   0.5128
 4   0.0000   0.0000  -0.3133  -0.0994
 Basis of invariant subspace
          1        2
 1   0.6551   0.1211
 2   0.5236   0.3286
 3  -0.5362   0.5974
 4   0.0956   0.7215

 Condition number estimate of the selected cluster of eigenvalues =   1.75e+00

 Condition number estimate of the specified invariant subspace =   3.22e+00
```