NAG 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

```
#include <nag.h>
#include <nagf08.h>
void nag_dtrsen (Nag_OrderType order, Nag_JobType job,
     Nag_ComputeQType compq, const Nag_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 $A = QTQ^{\mathrm{T}}$, 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: $\tilde{T} = Z^{\mathrm{T}}TZ$. Optionally the updated matrix $\tilde{Q}$ of Schur vectors is computed as $\tilde{Q} = QZ$, giving $A = \tilde{Q}\tilde{T}\tilde{Q}^{\mathrm{T}}$.

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 sub-matrix 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 $AQ_1 = Q_1T_{11}$, 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

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

5 Arguments

1: **order** – Nag_OrderType *Input*

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 2.3.1.3 in How to Use the NAG Library and its Documentation 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 $\max(1, \mathbf{n})$.

On entry: the eigenvalues in the selected cluster. To select a real eigenvalue λ_j, **select**$[j-1]$ must be set Nag_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 Nag_TRUE. A complex conjugate pair of eigenvalues **must** be either both included in the cluster or both excluded. See also Section 9.

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 $\max(1, \mathbf{pdt} \times \mathbf{n})$.

The (i,j)th element of the matrix T is stored in

t$[(j-1) \times \mathbf{pdt} + i - 1]$ when **order** = Nag_ColMajor;
t$[(i-1) \times \mathbf{pdt} + j - 1]$ when **order** = Nag_RowMajor.

On entry: the n by n upper quasi-triangular matrix T in canonical Schur form, as returned by nag_dhseqr (f08pec). See also Section 9.

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

8: **q**[*dim*] – double *Input/Output*

Note: the dimension, *dim*, of the array **q** must be at least

max(1, **pdq** × **n**) when **compq** = Nag_UpdateSchur;
1 when **compq** = Nag_NotQ.

The (i, j)th element of the matrix Q is stored in

q[$(j-1) \times$ **pdq** $+ i - 1$] when **order** = Nag_ColMajor;
q[$(i-1) \times$ **pdq** $+ j - 1$] when **order** = Nag_RowMajor.

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.

If **compq** = Nag_NotQ, **q** is not referenced.

9: **pdq** – Integer *Input*

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

Constraints:

if **compq** = Nag_UpdateSchur, **pdq** ≥ max(1, **n**);
if **compq** = Nag_NotQ, **pdq** ≥ 1.

10: **wr**[*dim*] – double *Output*
11: **wi**[*dim*] – double *Output*

Note: the dimension, *dim*, of the arrays **wr** and **wi** must be at least max(1, **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 9 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**.**code** = NE_REORDER (see Section 6), **s** is set to zero.

If **job** = Nag_DoNothing or Nag_Subspace, **s** is not referenced.

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**.**code** = NE_REORDER (see Section 6), **sep** is set to zero.

If **job** = Nag_DoNothing or Nag_EigVals, **sep** is not referenced.

15: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 2.3.1.2 in How to Use the NAG Library and its Documentation 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** $\geq \max(1, \mathbf{n})$;
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** $\geq \max(1, \mathbf{n})$.

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 2.7.6 in How to Use the NAG Library and its Documentation for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

NE_REORDER

The reordering of T failed because a selected eigenvalue was too close to an unselected eigenvalue.

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.*

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 reordering, 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, i.e., for a pair of complex eigenvalues to become purely real. The values of real eigenvalues however are never changed by the reordering.

8 Parallelism and Performance

nag_dtrsen (f08qgc) 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.

9 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).

10 Example

This example reorders the Schur factorization of the matrix $A = QTQ^{\mathrm{T}}$ 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 example program for nag_dtrsen (f08qgc) illustrates the computation of error bounds for the eigenvalues.

The original matrix A is given in Section 10 in nag_dorghr (f08nfc).

10.1 Program Text

```
/* nag_dtrsen (f08qgc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

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

int main(void)
{
  /* Scalars */
  Integer i, j, m, n, pda, pdc, pdq, pdt, select_len, w_len;
  Integer exit_status = 0;
  double alpha, beta, norm, s, sep;
  NagError fail;
  Nag_OrderType order;
  /* Arrays */
  double *a = 0, *c = 0, *q = 0, *t = 0, *wi = 0, *wr = 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_dtrsen (f08qgc) 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 NAG_COLUMN_MAJOR
  pda = n;
  pdc = n;
  pdq = n;
  pdt = n;
#else
  pda = n;
  pdc = n;
  pdq = n;
  pdt = n;
#endif
  w_len = n;
  select_len = n;

  /* Allocate memory */
  if (!(a = NAG_ALLOC(n * n, double)) ||
      !(c = NAG_ALLOC(n * n, double)) ||
      !(q = NAG_ALLOC(n * n, double)) ||
      !(wi = NAG_ALLOC(w_len, double)) ||
      !(wr = NAG_ALLOC(w_len, double)) ||
      !(select = NAG_ALLOC(select_len, Nag_Boolean)) ||
      !(t = NAG_ALLOC(n * n, double)))
  {
    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", &T(i, j));
#else
      scanf("%lf", &T(i, j));
#endif
  }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
  for (i = 1; i <= n; ++i) {
    for (j = 1; j <= n; ++j)
#ifdef _WIN32
      scanf_s("%lf", &Q(i, j));
#else
      scanf("%lf", &Q(i, j));
#endif
  }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
  for (i = 0; i < n; ++i) {
#ifdef _WIN32
    scanf_s("%39s", nag_enum_arg, (unsigned)_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);
  }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
```

```
  /* nag_dgemm (f16yac): Compute Q*T*QT and store in matrix A */
  alpha = 1.0;
  beta = 0.0;
  nag_dgemm(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_dgemm (f16yac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }
  nag_dgemm(order, Nag_NoTrans, Nag_Trans, n, n, n, alpha, c, pdc, q,
            pdq, beta, a, pda, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_dgemm (f16yac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  /* nag_gen_real_mat_print (x04cac): Print Matrix A. */
  fflush(stdout);
  nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, n,
                         a, pda, "Matrix A", 0, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_gen_real_mat_print (x04cac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }
  printf("\n");

  /* Reorder the Schur factorization T */
  /* nag_dtrsen (f08qgc).
   * Reorder Schur factorization of real matrix, form
   * orthonormal basis of right invariant subspace for
   * selected eigenvalues, with estimates of sensitivities
   */
  nag_dtrsen(order, Nag_DoBoth, Nag_UpdateSchur, select, n, t, pdt,
             q, pdq, wr, wi, &m, &s, &sep, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_dtrsen (f08qgc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  /* nag_dgemm (f16yac): Compute A = Q*T*Q^T - Qt*Tt*Qt^T */
  alpha = 1.0;
  beta = 0.0;
  nag_dgemm(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_dgemm (f16yac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }
  alpha = -1.0;
  beta = 1.0;
  nag_dgemm(order, Nag_NoTrans, Nag_Trans, n, n, n, alpha, c, pdc, q,
            pdq, beta, a, pda, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_dgemm (f16yac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  /* nag_dge_norm (f16rac): Find norm of matrix A and print warning if */
  /* it is too large */
  nag_dge_norm(order, Nag_OneNorm, n, n, a, pda, &norm, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_dge_norm (f16rac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
```

```
  }
  if (norm > pow(x02ajc(), 0.8)) {
    printf("%s\n%s\n",
           "Norm of Q*T*Q^H - (Qt*Tt*Qt^H) is much greater than 0.",
           "Schur factorization has failed.");
  }
  else {
    /* Print condition number estimates */
    printf(" Condition number estimate of the selected cluster of"
           " eigenvalues = %11.2e\n", 1.0 / s);
    printf("\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(wi);
  NAG_FREE(wr);
  NAG_FREE(select);

  return exit_status;
}
```

10.2 Program Data

```
nag_dtrsen (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
  Nag_TRUE Nag_FALSE Nag_FALSE Nag_TRUE      :End of select
```

10.3 Program Results

```
nag_dtrsen (f08qgc) Example Program Results

 Matrix A
         1       2       3       4
 1   0.3500  0.4500 -0.1400 -0.1700
 2   0.0900  0.0700 -0.5399  0.3500
 3  -0.4400 -0.3300 -0.0300  0.1700
 4   0.2500 -0.3200 -0.1300  0.1100

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

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