NAG C Library Function Document

nag_dtrsna (f08qlc)

1 Purpose

nag_dtrsna (f08qlc) estimates condition numbers for specified eigenvalues and/or right eigenvectors of a real upper quasi-triangular matrix.

2 Specification

```
void nag_dtrsna (Nag_OrderType order, Nag_JobType job, Nag_HowManyType how_many,
      const Boolean select[], Integer n, const double t[], Integer pdt,
      const double vl[], Integer pdvl, const double vr[], Integer pdvr, double s[],
      double sep[], Integer mm, Integer *m, NagError *fail)
```

3 Description

nag_dtrsna (f08qlc) estimates condition numbers for specified eigenvalues and/or right eigenvectors of a real upper quasi-triangular matrix T in canonical Schur form. These are the same as the condition numbers of the eigenvalues and right eigenvectors of an original matrix $A = ZTZ^T$ (with orthogonal Z), from which T may have been derived.

nag_dtrsna (f08qlc) computes the reciprocal of the condition number of an eigenvalue λ_i as

$$s_i = \frac{|v^H u|}{\|u\|_E \, \|v\|_E},$$

where u and v are the right and left eigenvectors of T, respectively, corresponding to λ_i. This reciprocal condition number always lies between zero (i.e., ill-conditioned) and one (i.e., well-conditioned).

An approximate error estimate for a computed eigenvalue λ_i is then given by

$$\frac{\epsilon\|T\|}{s_i},$$

where ϵ is the ***machine precision***.

To estimate the reciprocal of the condition number of the right eigenvector corresponding to λ_i, the function first calls nag_dtrexc (f08qfc) to reorder the eigenvalues so that λ_i is in the leading position:

$$T = Q \begin{pmatrix} \lambda_i & c^T \\ 0 & T_{22} \end{pmatrix} Q^T.$$

The reciprocal condition number of the eigenvector is then estimated as sep_i, the smallest singular value of the matrix $(T_{22} - \lambda_i I)$. This number ranges from zero (i.e., ill-conditioned) to very large (i.e., well-conditioned).

An approximate error estimate for a computed right eigenvector u corresponding to λ_i is then given by

$$\frac{\epsilon\|T\|}{sep_i}.$$

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 eigenvalues and/or eigenvectors, as follows:

if **job** = **Nag_EigVals**, then condition numbers for eigenvalues only are computed;

if **job** = **Nag_EigVecs**, then condition numbers for eigenvectors only are computed;

if **job** = **Nag_DoBoth**, then condition numbers for both eigenvalues and eigenvectors are computed.

Constraint: **job** = **Nag_EigVals**, **Nag_EigVecs** or **Nag_DoBoth**.

3: **how_many** – Nag_HowManyType *Input*

On entry: indicates how many condition numbers are to be computed, as follows:

if **how_many** = **Nag_ComputeAll**, then condition numbers for all eigenpairs are computed;

if **how_many** = **Nag_ComputeSelected**, then condition numbers for selected eigenpairs (as specified by **select**) are computed.

Constraint: **how_many** = **Nag_ComputeAll** or **Nag_ComputeSelected**.

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

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

On entry: **select** specifies the eigenpairs for which condition numbers are to be computed if **how_many** = **Nag_ComputeSelected**. To select condition numbers for the eigenpair corresponding to the real eigenvalue λ_j, **select**$[j]$ must be set **TRUE**. To select condition numbers corresponding to a complex conjugate pair of eigenvalues λ_j and λ_{j+1}, **select**$[j]$ and/or **select**$[j+1]$ must be set to **TRUE**.

select is not referenced if **how_many** = **Nag_ComputeAll**.

5: **n** – Integer *Input*

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

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

6: **t**$[dim]$ – const double *Input*

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 quasi-triangular matrix T in canonical Schur form, as returned by nag_dhseqr (f08pec).

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: **vl**[dim] – const double *Input*

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

$\max(1, \mathbf{pdvl} \times \mathbf{mm})$ when **job** = **Nag_EigVals** or **Nag_DoBoth** and **order** = **Nag_ColMajor**;

$\max(1, \mathbf{pdvl} \times \mathbf{n})$ when **job** = **Nag_EigVals** or **Nag_DoBoth** and **order** = **Nag_RowMajor**;

1 when **job** = **Nag_EigVecs**.

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

On entry: if **job** = **Nag_EigVals** or **Nag_DoBoth**, **vl** must contain the left eigenvectors of T (or of any matrix QTQ^T with Q orthogonal) corresponding to the eigenpairs specified by **how_many** and **select**. The eigenvectors **must** be stored in consecutive rows or columns of **vl**, as returned by nag_dtrevc (f08qkc) or nag_dhsein (f08pkc).

vl is not referenced if **job** = **Nag_EigVecs**.

9: **pdvl** – Integer *Input*

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

Constraints:

if **order** = **Nag_ColMajor**,
  if **job** = **Nag_EigVals** or **Nag_DoBoth**, **pdvl** $\geq \max(1, \mathbf{n})$;
  if **job** = **Nag_EigVecs**, **pdvl** ≥ 1;

if **order** = **Nag_RowMajor**,
  if **job** = **Nag_EigVals** or **Nag_DoBoth**, **pdvl** $\geq \max(1, \mathbf{mm})$;
  if **job** = **Nag_EigVecs**, **pdvl** ≥ 1.

10: **vr**[dim] – const double *Input*

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

$\max(1, \mathbf{pdvr} \times \mathbf{mm})$ when **job** = **Nag_EigVals** or **Nag_DoBoth** and **order** = **Nag_ColMajor**;

$\max(1, \mathbf{pdvr} \times \mathbf{n})$ when **job** = **Nag_EigVals** or **Nag_DoBoth** and **order** = **Nag_RowMajor**;

1 when **job** = **Nag_EigVecs**.

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

On entry: if **job** = **Nag_EigVals** or **Nag_DoBoth**, **vr** must contain the right eigenvectors of T (or of any matrix QTQ^T with Q orthogonal) corresponding to the eigenpairs specified by **how_many** and **select**. The eigenvectors **must** be stored in consecutive rows or columns of **vr**, as returned by nag_dtrevc (f08qkc) or nag_dhsein (f08pkc).

vr is not referenced if **job** = **Nag_EigVecs**.

11: **pdvr** – Integer *Input*

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

Constraints:

if **order** = **Nag_ColMajor**,
if **job** = **Nag_EigVals** or **Nag_DoBoth**, **pdvr** $\geq$ max(1, **n**);
if **job** = **Nag_EigVecs**, **pdvr** $\geq$ 1;

if **order** = **Nag_RowMajor**,
if **job** = **Nag_EigVals** or **Nag_DoBoth**, **pdvr** $\geq$ max(1, **mm**);
if **job** = **Nag_EigVecs**, **pdvr** $\geq$ 1.

12: **s**[dim] – double *Output*

Note: the dimension, dim, of the array **s** must be at least max(1, **mm**) when **job** = **Nag_EigVals** or **Nag_DoBoth** and at least 1 when **job** = **Nag_EigVecs**.

On exit: the reciprocal condition numbers of the selected eigenvalues if **job** = **Nag_EigVals** or **Nag_DoBoth**, stored in consecutive elements of the array. Thus **s**[j], **sep**[j] and the jth rows or columns of **vl** and **vr** all correspond to the same eigenpair (but not in general the jth eigenpair unless all eigenpairs have been selected). For a complex conjugate pair of eigenvalues, two consecutive elements of **s** are set to the same value.

s is not referenced if **job** = **Nag_EigVecs**.

13: **sep**[dim] – double *Output*

Note: the dimension, dim, of the array **sep** must be at least max(1, **mm**) when **job** = **Nag_EigVecs** or **Nag_DoBoth** and at least 1 when **job** = **Nag_EigVals**.

On exit: the estimated reciprocal condition numbers of the selected right eigenvectors if **job** = **Nag_EigVecs** or **Nag_DoBoth**, stored in consecutive elements of the array. For a complex eigenvector, two consecutive elements of **sep** are set to the same value. If the eigenvalues cannot be reordered to compute **sep**[j], then **sep**[j] is set to zero; this can only occur when the true value would be very small anyway.

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

14: **mm** – Integer *Input*

On entry: the number of elements in the arrays **s** and **sep**, and the number of rows or columns (depending on the value of **order**) in the arrays **vl** and **vr** (if used). The precise number required, m, is n if **how_many** = **Nag_ComputeAll**; if **how_many** = **Nag_ComputeSelected**, m is obtained by counting 1 for each selected real eigenvalue, and 2 for each selected complex conjugate pair of eigenvalues (see **select**), in which case $0 \leq m \leq n$.

Constraint: **mm** $\geq$ **m**.

15: **m** – Integer * *Output*

On exit: m, the number of elements of **s** and/or **sep** actually used to store the estimated condition numbers. If **how_many** = **Nag_ComputeAll**, **m** is set to n.

16: **fail** – NagError * *Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** $\geq$ 0.

On entry, **pdt** = ⟨*value*⟩.
Constraint: **pdt** > 0.

On entry, **pdvl** = ⟨*value*⟩.
Constraint: **pdvl** > 0.

On entry, **pdvr** = ⟨*value*⟩.
Constraint: **pdvr** > 0.

NE_INT_2

On entry, **mm** = ⟨*value*⟩, **m** = ⟨*value*⟩.
Constraint: **mm** $\geq$ **m**.

On entry, **pdt** = ⟨*value*⟩, **n** = ⟨*value*⟩.
Constraint: **pdt** $\geq \max(1, \mathbf{n})$.

NE_ENUM_INT_2

On entry, **job** = ⟨*value*⟩, **n** = ⟨*value*⟩, **pdvl** = ⟨*value*⟩.
Constraint: if **job** = **Nag_EigVals** or **Nag_DoBoth**, **pdvl** $\geq \max(1, \mathbf{n})$;
if **job** = **Nag_EigVecs**, **pdvl** ≥ 1.

On entry, **job** = ⟨*value*⟩, **n** = ⟨*value*⟩, **pdvr** = ⟨*value*⟩.
Constraint: if **job** = **Nag_EigVals** or **Nag_DoBoth**, **pdvr** $\geq \max(1, \mathbf{n})$;
if **job** = **Nag_EigVecs**, **pdvr** ≥ 1.

On entry, **job** = ⟨*value*⟩, **mm** = ⟨*value*⟩, **pdvl** = ⟨*value*⟩.
Constraint: if **job** = **Nag_EigVals** or **Nag_DoBoth**, **pdvl** $\geq \max(1, \mathbf{mm})$;
if **job** = **Nag_EigVecs**, **pdvl** ≥ 1.

On entry, **job** = ⟨*value*⟩, **mm** = ⟨*value*⟩, **pdvr** = ⟨*value*⟩.
Constraint: if **job** = **Nag_EigVals** or **Nag_DoBoth**, **pdvr** $\geq \max(1, \mathbf{mm})$;
if **job** = **Nag_EigVecs**, **pdvr** ≥ 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 values sep_i may over estimate the true value, but seldom by a factor of more than 3.

8 Further Comments

For a description of canonical Schur form, see the document for nag_dhseqr (f08pec).

The complex analogue of this function is nag_ztrsna (f08qyc).

9 Example

To compute approximate error estimates for all the eigenvalues and right eigenvectors of the matrix T, where

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

9.1 Program Text

```
/* nag_dtrsna (f08qlc) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

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

int main(void)
{
  /* Scalars */
  Integer  i, j, m, n, pdt, pdvl, pdvr;
  Integer  select_len, s_len;
  Integer  exit_status=0;
  double   eps, tnorm;
  NagError fail;
  Nag_OrderType order;
  /* Arrays */
  double  *s=0, *sep=0, *t=0, *vl=0, *vr=0;
  Boolean *select=0;

#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);
  Vprintf("f08qlc Example Program Results\n");

  /* Skip heading in data file */
  Vscanf("%*[^\n] ");
  Vscanf("%ld%*[^\n] ", &n);
#ifdef NAG_COLUMN_MAJOR
  pdt = n;
  pdvl = n;
  pdvr = n;
#else
  pdt = n;
  pdvl = n;
  pdvr = n;
#endif
  select_len = 1;
  s_len = n;

  /* Allocate memory */
  if ( !(t = NAG_ALLOC(n * n, double)) ||
       !(vl = NAG_ALLOC(n * n, double)) ||
       !(vr = NAG_ALLOC(n * n, double)) ||
       !(s = NAG_ALLOC(s_len, double)) ||
       !(sep = NAG_ALLOC(s_len, double)) ||
       !(select = NAG_ALLOC(select_len, Boolean)) )
    {
```

```
      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] ");

  /* Calculate right and left eigrnvectors of T */
  f08qkc(order, Nag_BothSides, Nag_ComputeAll, select, n, t, pdt,
         vl, pdvl, vr, pdvr, n, &m, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from f08qkc.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  /* Estimate condition numbers for all the eigenvalues and */
  /* right eigenvectors of T */
  f08qlc(order, Nag_DoBoth, Nag_ComputeAll, select, n, t, pdt,
         vl, pdvl, vr, pdvr, s, sep, n, &m, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from f08qlc.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  /* Print condition numbers of eigenvalues and right eigenvectors */
  Vprintf("\nS\n");
  for (i = 0; i < n; ++i)
    Vprintf("%11.1e", s[i]);
  Vprintf("\n\nSep\n");
  for (i = 0; i < n; ++i)
    Vprintf("%11.1e", sep[i]);
  Vprintf("\n");
  /* Calculate approximate error estimates (using the 1-norm) */
  f16rac(order, Nag_OneNorm, n, n, t, pdt, &tnorm, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from f16rac.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  eps = X02AJC;
  Vprintf("\nApproximate error estimates for eigenvalues"
          "of T (machine dependent)\n");
  for (i = 0; i < m; ++i)
    Vprintf("%11.1e", eps*tnorm/s[i]);
  Vprintf("\n\nApproximate error estimates for right eigenvectors"
          "of T (machine dependent)\n");
  for (i = 0; i < m; ++i)
    Vprintf("%11.1e", eps*tnorm/sep[i]);
  Vprintf("\n");
 END:
  if (t) NAG_FREE(t);
  if (s) NAG_FREE(s);
  if (sep) NAG_FREE(sep);
  if (vl) NAG_FREE(vl);
  if (vr) NAG_FREE(vr);
  if (select) NAG_FREE(select);

  return exit_status;
}
```

9.2 Program Data

```
f08qlc 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
```

9.3 Program Results

```
f08qlc Example Program Results

S
    9.9e-01    7.0e-01    7.0e-01    5.7e-01

Sep
    6.3e-01    3.7e-01    3.7e-01    3.1e-01

Approximate error estimates for eigenvaluesof T (machine dependent)
    9.6e-17    1.4e-16    1.4e-16    1.7e-16

Approximate error estimates for right eigenvectorsof T (machine dependent)
    1.5e-16    2.6e-16    2.6e-16    3.1e-16
```