

# NAG 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

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

void nag_dtrsna (Nag_OrderType order, Nag_JobType job,
    Nag_HowManyType how_many, const Nag_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  $\lambda_i$  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  $\lambda_i$ . 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  $\lambda_i$  is then given by

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

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

To estimate the reciprocal of the condition number of the right eigenvector corresponding to  $\lambda_i$ , the function first calls nag\_dtrexc (f08qfc) to reorder the eigenvalues so that  $\lambda_i$  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  $\lambda_i$  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 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 eigenvalues and/or eigenvectors.  
**job** = Nag\_EigVals  
Condition numbers for eigenvalues only are computed.  
**job** = Nag\_EigVecs  
Condition numbers for eigenvectors only are computed.  
**job** = Nag\_DoBoth  
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.  
**how\_many** = Nag\_ComputeAll  
Condition numbers for all eigenpairs are computed.  
**how\_many** = Nag\_ComputeSelected  
Condition numbers for selected eigenpairs (as specified by **select**) are computed.  
*Constraint:* **how\_many** = Nag\_ComputeAll or Nag\_ComputeSelected.
- 4: **select**[*dim*] – const Nag\_Boolean *Input*  
**Note:** the dimension, *dim*, of the array **select** must be at least  
**n** when **how\_many** = Nag\_ComputeSelected;  
otherwise **select** may be **NULL**.  
*On entry:* 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  $\lambda_j$ , **select**[*j* – 1] must be set Nag\_TRUE. To select condition numbers corresponding to a complex conjugate pair of eigenvalues  $\lambda_j$  and  $\lambda_{j+1}$ , **select**[*j* – 1] and/or **select**[*j*] must be set to Nag\_TRUE.  
If **how\_many** = Nag\_ComputeAll, **select** is not referenced and may be **NULL**.
- 5: **n** – Integer *Input*  
*On entry:* *n*, the order of the matrix *T*.  
*Constraint:* **n** ≥ 0.
- 6: **t**[*dim*] – const double *Input*  
**Note:** the dimension, *dim*, of the array **t** must be at least **pdt** × **n**.  
The (*i*, *j*)th element of the matrix *T* is stored in  
**t**[(*j* – 1) × **pdt** + *i* – 1] when **order** = Nag\_ColMajor;  
**t**[(*i* – 1) × **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).

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: **vl**[*dim*] – const double *Input*

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

**pdvl**  $\times$  **mm** when **job** = Nag\_EigVals or Nag\_DoBoth and **order** = Nag\_ColMajor;  
**n**  $\times$  **pdvl** when **job** = Nag\_EigVals or Nag\_DoBoth and **order** = Nag\_RowMajor;  
 otherwise **vl** may be **NULL**.

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

**vl**[ $(j - 1) \times \mathbf{pdvl} + i - 1$ ] when **order** = Nag\_ColMajor;  
**vl**[ $(i - 1) \times \mathbf{pdvl} + j - 1$ ] when **order** = Nag\_RowMajor.

*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\_dhsein (f08pkc) or nag\_dtrevc (f08qkc).

If **job** = Nag\_EigVecs, **vl** is not referenced and may be **NULL**.

9: **pdvl** – Integer *Input*

*On entry:* the stride separating 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**  $\geq n$ ;  
     if **job** = Nag\_EigVecs, **vl** may be **NULL**.;  
 if **order** = Nag\_RowMajor,  
     if **job** = Nag\_EigVals or Nag\_DoBoth, **pdvl**  $\geq \mathbf{mm}$ ;  
     if **job** = Nag\_EigVecs, **vl** may be **NULL**..

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

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

**pdvr**  $\times$  **mm** when **job** = Nag\_EigVals or Nag\_DoBoth and **order** = Nag\_ColMajor;  
**n**  $\times$  **pdvr** when **job** = Nag\_EigVals or Nag\_DoBoth and **order** = Nag\_RowMajor;  
 otherwise **vr** may be **NULL**.

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

**vr**[ $(j - 1) \times \mathbf{pdvr} + i - 1$ ] when **order** = Nag\_ColMajor;  
**vr**[ $(i - 1) \times \mathbf{pdvr} + j - 1$ ] when **order** = Nag\_RowMajor.

*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\_dhsein (f08pkc) or nag\_dtrevc (f08qkc).

If **job** = Nag\_EigVecs, **vr** is not referenced and may be **NULL**.

- 11: **pdvr** – Integer *Input*
- On entry:* the stride separating 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**  $\geq$  **n**;
    - if **job** = Nag\_EigVecs, **vr** may be **NULL**;
  - if **order** = Nag\_RowMajor,
    - if **job** = Nag\_EigVals or Nag\_DoBoth, **pdvr**  $\geq$  **mm**;
    - if **job** = Nag\_EigVecs, **vr** may be **NULL**.
- 12: **s**[*dim*] – double *Output*
- Note:** the dimension, *dim*, of the array **s** must be at least
- mm** when **job** = Nag\_EigVals or Nag\_DoBoth;
  - otherwise **s** may be **NULL**.
- 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* – 1], **sep**[*j* – 1] and the *j*th rows or columns of **vl** and **vr** all correspond to the same eigenpair (but not in general the *j*th 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.
- If **job** = Nag\_EigVecs, **s** is not referenced and may be **NULL**.
- 13: **sep**[*dim*] – double *Output*
- Note:** the dimension, *dim*, of the array **sep** must be at least
- mm** when **job** = Nag\_EigVecs or Nag\_DoBoth;
  - otherwise **sep** may be **NULL**.
- 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.
- If **job** = Nag\_EigVals, **sep** is not referenced and may be **NULL**.
- 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**.
- Constraint:* if **how\_many** = Nag\_ComputeAll, **mm**  $\geq$  **n**.
- 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 \* *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  $\langle value \rangle$  had an illegal value.

### NE\_ENUM\_INT\_2

On entry, **how\_many** =  $\langle value \rangle$ , **mm** =  $\langle value \rangle$  and **n** =  $\langle value \rangle$ .

Constraint: if **how\_many** = Nag\_ComputeAll, **mm**  $\geq$  **n**.

On entry, **job** =  $\langle value \rangle$ , **pdvl** =  $\langle value \rangle$ , **mm** =  $\langle value \rangle$ .

Constraint: if **job** = Nag\_EigVals or Nag\_DoBoth, **pdvl**  $\geq$  **mm**.

On entry, **job** =  $\langle value \rangle$ , **pdvl** =  $\langle value \rangle$  and **n** =  $\langle value \rangle$ .

Constraint: if **job** = Nag\_EigVals or Nag\_DoBoth, **pdvl**  $\geq$  **n**.

On entry, **job** =  $\langle value \rangle$ , **pdvr** =  $\langle value \rangle$ , **mm** =  $\langle value \rangle$ .

Constraint: if **job** = Nag\_EigVals or Nag\_DoBoth, **pdvr**  $\geq$  **mm**.

On entry, **job** =  $\langle value \rangle$ , **pdvr** =  $\langle value \rangle$  and **n** =  $\langle value \rangle$ .

Constraint: if **job** = Nag\_EigVals or Nag\_DoBoth, **pdvr**  $\geq$  **n**.

### NE\_INT

On entry, **n** =  $\langle value \rangle$ .

Constraint: **n**  $\geq$  0.

On entry, **pdt** =  $\langle value \rangle$ .

Constraint: **pdt**  $>$  0.

On entry, **pdvl** =  $\langle value \rangle$ .

Constraint: **pdvl**  $>$  0.

On entry, **pdvr** =  $\langle value \rangle$ .

Constraint: **pdvr**  $>$  0.

### NE\_INT\_2

On entry, **mm** =  $\langle value \rangle$  and **m** =  $\langle value \rangle$ .

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

On entry, **pdt** =  $\langle value \rangle$  and **n** =  $\langle value \rangle$ .

Constraint: **pdt**  $\geq$  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 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.

## 7 Accuracy

The computed values  $sep_i$  may over estimate the true value, but seldom by a factor of more than 3.

## 8 Parallelism and Performance

nag\_dtrsna (f08qlc) 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

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

The complex analogue of this function is nag\_ztrsna (f08qyc).

## 10 Example

This example computes approximate error estimates for all the eigenvalues and right eigenvectors of the matrix  $T$ , where

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

### 10.1 Program Text

```
/* nag_dtrsna (f08qlc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#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 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;

#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]
#endif
```

```

    order = Nag_RowMajor;
#endif

    INIT_FAIL(fail);

    printf("nag_dtrsna (f08qlc) Example Program Results\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
    pdt = n;
    pdvl = n;
    pdvr = n;
#else
    pdt = n;
    pdvl = n;
    pdvr = n;
#endif
    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)))
    {
        printf("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)
#ifdef _WIN32
            scanf_s("%lf", &T(i, j));
#else
            scanf("%lf", &T(i, j));
#endif
    }
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif

    /* Calculate right and left eigenvectors of real upper quasi-triangular
     * matrix T using nag_dtrevc (f08qkc).
     */
    nag_dtrevc(order, Nag_BothSides, Nag_ComputeAll, NULL, n, t, pdt,
               vl, pdvl, vr, pdvr, n, &m, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_dtrevc (f08qkc).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

    /* Estimate condition numbers for all the eigenvalues and
     * right eigenvectors of real upper quasi-triangular matrix T using
     * nag_dtrsna (f08qlc).
     */
    nag_dtrsna(order, Nag_DoBoth, Nag_ComputeAll, NULL, n, t, pdt,

```

```

        vl, pdvl, vr, pdvr, s, sep, n, &m, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_dtrsna (f08qlc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print condition numbers of eigenvalues and right eigenvectors */
printf("\nS\n");
for (i = 0; i < n; ++i)
    printf("%11.1e", s[i]);
printf("\n\nSep\n");
for (i = 0; i < n; ++i)
    printf("%11.1e", sep[i]);
printf("\n");
/* Calculate approximate error estimates which depends on the 1-norm
 * of matrix T. The 1-norm of T is calculated using nag_dge_norm (f16rac).
 */
nag_dge_norm(order, Nag_OneNorm, n, n, t, pdt, &tnorm, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_dge_norm (f16rac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* error estimates also depend on nag_machine_precision (x02ajc). */
eps = nag_machine_precision;
printf("\nApproximate error estimates for eigenvalues"
      "of T (machine dependent)\n");
for (i = 0; i < m; ++i)
    printf("%11.1e", eps * tnorm / s[i]);
printf("\n\nApproximate error estimates for right eigenvectors"
      "of T (machine dependent)\n");
for (i = 0; i < m; ++i)
    printf("%11.1e", eps * tnorm / sep[i]);
printf("\n");
END:
    NAG_FREE(t);
    NAG_FREE(s);
    NAG_FREE(sep);
    NAG_FREE(vl);
    NAG_FREE(vr);

    return exit_status;
}

```

## 10.2 Program Data

```

nag_dtrsna (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

```

## 10.3 Program Results

```

nag_dtrsna (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 eigenvalues of  $T$  (machine dependent)

9.6e-17	1.4e-16	1.4e-16	1.7e-16
---------	---------	---------	---------

Approximate error estimates for right eigenvectors of  $T$  (machine dependent)

1.5e-16	2.6e-16	2.6e-16	3.1e-16
---------	---------	---------	---------

---