NAG Library Function Document

nag_ztrsna (f08qyc)

1 Purpose

nag_ztrsna (f08qyc) estimates condition numbers for specified eigenvalues and/or right eigenvectors of a complex upper triangular matrix.

2 Specification

```
#include <nag.h>
#include <nagf08.h>
void nag_ztrsna (Nag_OrderType order, Nag_JobType job,
     Nag_HowManyType how_many, const Nag_Boolean select[], Integer n,
     const Complex t[], Integer pdt, const Complex vl[], Integer pdvl,
     const Complex vr[], Integer pdvr, double s[], double sep[], Integer mm,
     Integer *m, NagError *fail)
```

3 Description

nag_ztrsna (f08qyc) estimates condition numbers for specified eigenvalues and/or right eigenvectors of a complex upper triangular matrix T. These are the same as the condition numbers of the eigenvalues and right eigenvectors of an original matrix $A = ZTZ^{\mathrm{H}}$ (with unitary Z), from which T may have been derived.

nag_ztrsna (f08qyc) computes the reciprocal of the condition number of an eigenvalue λ_i as

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

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_ztrexc (f08qtc) to reorder the eigenvalues so that λ_i is in the leading position:

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

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 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 eigenvalue λ_j, **select**[$j-1$] 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 Complex *Input*

Note: the dimension, *dim*, of the array **t** must be at least **pdt** $\times$ **n**.

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

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

On entry: the n by n upper triangular matrix T, as returned by nag_zhseqr (f08psc).

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

8: **vl**[dim] – const Complex *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[$(j-1) \times$ **pdvl** $+ i - 1$] when **order** = Nag_ColMajor;
vl[$(i-1) \times$ **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^{H} with Q unitary) corresponding to the eigenpairs specified by **how_many** and **select**. The eigenvectors **must** be stored in consecutive rows or columns (depending on the value of **order**) of **vl**, as returned by nag_zhsein (f08pxc) or nag_ztrevc (f08qxc).

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** $\geq$ **mm**;
if **job** = Nag_EigVecs, **vl** may be **NULL**..

10: **vr**[dim] – const Complex *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[$(j-1) \times$ **pdvr** $+ i - 1$] when **order** = Nag_ColMajor;
vr[$(i-1) \times$ **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^{H} with Q unitary) corresponding to the eigenpairs specified by **how_many** and **select**. The eigenvectors **must** be stored in consecutive rows or columns (depending on the value of **order**) of **vr**, as returned by nag_zhsein (f08pxc) or nag_ztrevc (f08qxc).

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 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).

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.

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, $required_rowcol$, is n if **how_many** = Nag_ComputeAll; if **how_many** = Nag_ComputeSelected, $required_rowcol$ is the number of selected eigenpairs (see **select**), in which case $0 \leq required_rowcol \leq n$.

Constraints:

if **how_many** = Nag_ComputeAll, **mm** $\geq$ **n**;
otherwise **mm** $\geq required_rowcol$.

15: **m** – Integer * *Output*

On exit: $required_rowcol$, the number of selected eigenpairs. 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 ⟨*value*⟩ had an illegal value.

NE_ENUM_INT_2

On entry, **how_many** = ⟨*value*⟩, **mm** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: if **how_many** = Nag_ComputeAll, **mm** ≥ **n**;
otherwise **mm** ≥ $required_rowcol$.

On entry, **job** = ⟨*value*⟩, **pdvl** = ⟨*value*⟩, **mm** = ⟨*value*⟩.
Constraint: if **job** = Nag_EigVals or Nag_DoBoth, **pdvl** ≥ **mm**.

On entry, **job** = ⟨*value*⟩, **pdvl** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: if **job** = Nag_EigVals or Nag_DoBoth, **pdvl** ≥ **n**.

On entry, **job** = ⟨*value*⟩, **pdvr** = ⟨*value*⟩, **mm** = ⟨*value*⟩.
Constraint: if **job** = Nag_EigVals or Nag_DoBoth, **pdvr** ≥ **mm**.

On entry, **job** = ⟨*value*⟩, **pdvr** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: if **job** = Nag_EigVals or Nag_DoBoth, **pdvr** ≥ **n**.

NE_INT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 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, **pdt** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: **pdt** ≥ 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_ztrsna (f08qyc) 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 real analogue of this function is nag_dtrsna (f08qlc).

10 Example

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

$$T = \begin{pmatrix} -6.0004 - 6.9999i & 0.3637 - 0.3656i & -0.1880 + 0.4787i & 0.8785 - 0.2539i \\ 0.0000 + 0.0000i & -5.0000 + 2.0060i & -0.0307 - 0.7217i & -0.2290 + 0.1313i \\ 0.0000 + 0.0000i & 0.0000 + 0.0000i & 7.9982 - 0.9964i & 0.9357 + 0.5359i \\ 0.0000 + 0.0000i & 0.0000 + 0.0000i & 0.0000 + 0.0000i & 3.0023 - 3.9998i \end{pmatrix}.$$

10.1 Program Text

```
/* nag_ztrsna (f08qyc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <naga02.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;
  Complex *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]
  order = Nag_RowMajor;
#endif

  INIT_FAIL(fail);
```

```
  printf("nag_ztrsna (f08qyc) 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, Complex)) ||
      !(vl = NAG_ALLOC(n * n, Complex)) ||
      !(vr = NAG_ALLOC(n * n, Complex)) ||
      !(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 , %lf ) ", &T(i, j).re, &T(i, j).im);
#else
      scanf(" ( %lf , %lf ) ", &T(i, j).re, &T(i, j).im);
#endif
  }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  /* Calculate right and left eigrnvectors of T */
  /* nag_ztrevc (f08qxc).
   * Left and right eigenvectors of complex upper triangular
   * matrix
   */
  nag_ztrevc(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_ztrevc (f08qxc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  /* Estimate condition numbers for all the eigenvalues and */
  /* right eigenvectors of T */
  /* nag_ztrsna (f08qyc).
   * Estimates of sensitivities of selected eigenvalues and
   * eigenvectors of complex upper triangular matrix
   */
  nag_ztrsna(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_ztrsna (f08qyc).\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 (using the 1-norm) */
  /* nag_zge_norm (f16uac).
   * 1-norm, infinity-norm, Frobenius norm, largest absolute
   * element, complex general matrix
   */
  nag_zge_norm(order, Nag_OneNorm, n, n, t, pdt, &tnorm, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_zge_norm (f16uac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }
  /* nag_machine_precision (x02ajc).
   * The machine precision
   */
  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_ztrsna (f08qyc) 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
```

10.3 Program Results

```
nag_ztrsna (f08qyc) Example Program Results

S
    9.9e-01    1.0e+00    9.8e-01    9.8e-01

Sep
    8.4e+00    8.0e+00    5.8e+00    5.8e+00
```

```
Approximate error estimates for eigenvaluesof T (machine dependent)
    1.0e-15    1.0e-15    1.1e-15    1.1e-15

Approximate error estimates for right eigenvectorsof T (machine dependent)
    1.2e-16    1.3e-16    1.8e-16    1.8e-16
```