NAG Library Function Document

nag_ztgsna (f08yyc)

1 Purpose
nag_ztgsna (f08yyc) estimates condition numbers for specified eigenvalues and/or eigenvectors of a complex matrix pair in generalized Schur form.

2 Specification

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

void nag_ztgsna (Nag_OrderType order, Nag_JobType job,
                Nag_HowManyType how_many, const Nag_Boolean select[],
                Integer n, const Complex a[], Integer pda, const Complex b[],
                Integer pdb, const Complex vl[], Integer pdvl, const Complex vr[],
                Integer pdvr, double s[], double dif[], Integer mm, Integer *m,
                NagError *fail)
```

3 Description
nag_ztgsna (f08yyc) estimates condition numbers for specified eigenvalues and/or right eigenvectors of an \(n\) by \(n\) matrix pair \((S, T)\) in generalized Schur form. The function actually returns estimates of the reciprocals of the condition numbers in order to avoid possible overflow.

The pair \((S, T)\) are in generalized Schur form if \(S\) and \(T\) are upper triangular as returned, for example, by nag_zgges (f08xnc) or nag_zggesx (f08xpc), or nag_zhgeqz (f08xsc) with \(\text{job} = \text{Nag_Schur}\). The diagonal elements define the generalized eigenvalues \((\alpha_i, \beta_i)\), for \(i = 1, 2, \ldots, n\), of the pair \((S, T)\) and the eigenvalues are given by

\[
\lambda_i = \alpha_i / \beta_i,
\]

so that

\[
\beta_i S x_i = \alpha_i T x_i \quad \text{or} \quad S x_i = \lambda_i T x_i,
\]

where \(x_i\) is the corresponding (right) eigenvector.

If \(S\) and \(T\) are the result of a generalized Schur factorization of a matrix pair \((A, B)\)

\[
A = QSZ^H, \quad B = QTZ^H
\]

then the eigenvalues and condition numbers of the pair \((S, T)\) are the same as those of the pair \((A, B)\).

Let \((\alpha, \beta) \neq (0, 0)\) be a simple generalized eigenvalue of \((A, B)\). Then the reciprocal of the condition number of the eigenvalue \(\lambda = \alpha / \beta\) is defined as

\[
s(\lambda) = \left( \frac{|y^H Ax|^2 + |y^H Bx|^2}{\|x\|_2 \|y\|_2} \right)^{1/2},
\]

where \(x\) and \(y\) are the right and left eigenvectors of \((A, B)\) corresponding to \(\lambda\). If both \(\alpha\) and \(\beta\) are zero, then \((A, B)\) is singular and \(s(\lambda) = -1\) is returned.

If \(U\) and \(V\) are unitary transformations such that

\[
U^H (A, B) V = (S, T) = \begin{pmatrix} \alpha & * \\ 0 & S_{22} \end{pmatrix} \begin{pmatrix} \beta & * \\ 0 & T_{22} \end{pmatrix},
\]

where \(S_{22}\) and \(T_{22}\) are \((n-1)\) by \((n-1)\) matrices, then the reciprocal condition number is given by

\[
\text{Dif}(x) \equiv \text{Dif}(y) = \text{Dif}((\alpha, \beta), (S_{22}, T_{22})) = \sigma_{\min}(Z),
\]
where $\sigma_{\text{min}}(Z)$ denotes the smallest singular value of the $2(n-1) \times 2(n-1)$ matrix

$$Z = \begin{pmatrix} \alpha \otimes I & -1 \otimes S_{22} \\ \beta \otimes I & -1 \otimes T_{22} \end{pmatrix}$$

and $\otimes$ is the Kronecker product.

See Sections 2.4.8 and 4.11 of Anderson et al. (1999) and Kågström and Poromaa (1996) for further details and information.

4 References


5 Arguments

1: \texttt{order} – Nag_OrderType \hspace{1cm} \textit{Input}

On entry: the \texttt{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 \texttt{order = Nag_RowMajor}. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

Constraint: \texttt{order = Nag_RowMajor} or \texttt{Nag_ColMajor}.

2: \texttt{job} – Nag_JobType \hspace{1cm} \textit{Input}

On entry: indicates whether condition numbers are required for eigenvalues and/or eigenvectors.

\texttt{job = Nag_EigVals}

Condition numbers for eigenvalues only are computed.

\texttt{job = Nag_EigVecs}

Condition numbers for eigenvectors only are computed.

\texttt{job = Nag_DoBoth}

Condition numbers for both eigenvalues and eigenvectors are computed.

Constraint: \texttt{job = Nag_EigVals}, \texttt{Nag_EigVecs} or \texttt{Nag_DoBoth}.

3: \texttt{how_many} – Nag_HowManyType \hspace{1cm} \textit{Input}

On entry: indicates how many condition numbers are to be computed.

\texttt{how_many = Nag_ComputeAll}

Condition numbers for all eigenpairs are computed.

\texttt{how_many = Nag_ComputeSelected}

Condition numbers for selected eigenpairs (as specified by \texttt{select}) are computed.

Constraint: \texttt{how_many = Nag_ComputeAll} or \texttt{Nag_ComputeSelected}.

4: \texttt{select[dim]} – const Nag_Boolean \hspace{1cm} \textit{Input}

\textbf{Note:} the dimension, \texttt{dim}, of the array \texttt{select} must be at least \texttt{n} when \texttt{how_many = Nag_ComputeSelected}; otherwise \texttt{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 $\lambda_j$, 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

On entry: $n$, the order of the matrix pair $(S,T)$.

Constraint: $n \geq 0$.

6: $a[dim]$ – Complex

Note: the dimension, dim, of the array $a$ must be at least pda × n.

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

$$a[(j-1) \times \text{pda} + i - 1] \text{ when order = Nag_ColMajor;}$$

$$a[(i-1) \times \text{pda} + j - 1] \text{ when order = Nag_RowMajor.}$$

On entry: the upper triangular matrix $S$.

7: $pda$ – Integer

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

Constraint: $pda \geq n$.

8: $b[dim]$ – Complex

Note: the dimension, dim, of the array $b$ must be at least pdb × n.

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

$$b[(j-1) \times \text{pdb} + i - 1] \text{ when order = Nag_ColMajor;}$$

$$b[(i-1) \times \text{pdb} + j - 1] \text{ when order = Nag_RowMajor.}$$

On entry: the upper triangular matrix $T$.

9: $pdb$ – Integer

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

Constraint: $pdb \geq n$.

10: $vl[dim]$ – Complex

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

$$\text{pdvl} \times \text{mm} \text{ when job = Nag_EigVals or Nag_DoBoth and order = Nag_ColMajor;}$$

$$\text{n} \times \text{pdvl} \text{ when job = Nag_EigVals or Nag_DoBoth and order = Nag_RowMajor;}$$

otherwise $vl$ may be NULL.

The $i$th element of the $j$th vector is stored in

$$vl[(j-1) \times \text{pdvl} + i - 1] \text{ when order = Nag_ColMajor;}$$

$$vl[(i-1) \times \text{pdvl} + j - 1] \text{ when order = Nag_RowMajor.}$$

On entry: if job = Nag_EigVals or Nag_DoBoth, $vl$ must contain left eigenvectors of $(S,T)$, corresponding to the eigenpairs specified by how_many and select. The eigenvectors must be stored in consecutive columns of $vl$, as returned by nag_zggev (f08wnc) or nag_ztgevc (f08yxc).

If job = Nag_EigVecs, $vl$ is not referenced and may be NULL.
11: \( pdvl \) – Integer  

\textit{Input}

\textit{On entry:} the stride used in the array \( vl \).

\textit{Constraints:}

- if order = Nag_ColMajor,
  - if job = Nag_EigVals or Nag_DoBoth, \( pdvl \geq n \);
  - otherwise \( pdvl \geq 1 \);
- if order = Nag_RowMajor,
  - if job = Nag_EigVals or Nag_DoBoth, \( pdvl \geq mm \);
  - otherwise \( vl \) may be NULL.

12: \( vr[dim] \) – const Complex  

\textit{Input}

\textit{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 \( i \)th element of the \( j \)th vector is stored in

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

\textit{On entry:} if \( job = Nag\_EigVals \) or \( Nag\_DoBoth \), \( vr \) must contain right eigenvectors of \( (S,T) \), corresponding to the eigenpairs specified by \textit{how_many} and \textit{select}. The eigenvectors must be stored in consecutive columns of \( vr \), as returned by \textit{nag_zggev} (f08wnc) or \textit{nag_ztgevc} (f08yxc).

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

13: \( pdvr \) – Integer  

\textit{Input}

\textit{On entry:} the stride used in the array \( vr \).

\textit{Constraints:}

- if order = Nag_ColMajor,
  - if job = Nag_EigVals or Nag_DoBoth, \( pdvr \geq n \);
  - otherwise \( pdvr \geq 1 \);
- if order = Nag_RowMajor,
  - if job = Nag_EigVals or Nag_DoBoth, \( pdvr \geq mm \);
  - otherwise \( vr \) may be NULL.

14: \( s[dim] \) – double  

\textit{Output}

\textit{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.

\textit{On exit:} if \( job = Nag\_EigVals \) or \( Nag\_DoBoth \), the reciprocal condition numbers of the selected eigenvalues, stored in consecutive elements of the array.

If \( job = Nag\_EigVecs \), \( s \) is not referenced and may be NULL.

15: \( dif[dim] \) – double  

\textit{Output}

\textit{Note:} the dimension, \( dim \), of the array \( dif \) must be at least

\( mm \) when \( job = Nag\_EigVecs \) or \( Nag\_DoBoth \);
otherwise \( dif \) may be NULL.

\textit{On exit:} if \( job = Nag\_EigVecs \) or \( Nag\_DoBoth \), the estimated reciprocal condition numbers of the selected eigenvectors, stored in consecutive elements of the array. If the eigenvalues cannot be
reordered to compute $\text{dif}[j-1], \text{dif}[j-1]$ is set to 0; this can only occur when the true value would be very small anyway.

If $\text{job} = \text{Nag\_EigVals}$, $\text{dif}$ is not referenced and may be $\text{NULL}$.

16: $\text{mm} - \text{Integer}$

*Input*

On entry: the number of elements in the arrays $\text{s}$ and $\text{dif}$.

*Constraints:*

if $\text{how\_many} = \text{Nag\_ComputeAll}$, $\text{mm} \geq n$;

otherwise $\text{mm} \geq$ the number of selected eigenvalues.

17: $\text{m} - \text{Integer}$

*Output*

On exit: the number of elements of the arrays $\text{s}$ and $\text{dif}$ used to store the specified condition numbers; for each selected eigenvalue one element is used.

If $\text{how\_many} = \text{Nag\_ComputeAll}$, $\text{m}$ is set to $n$.

18: $\text{fail} - \text{NagError}$

*Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

**NE\_ALLOC\_FAIL**

Dynamic memory allocation failed.

See Section 3.2.1.2 in the Essential Introduction for further information.

**NE\_BAD\_PARAM**

On entry, argument $\langle value \rangle$ had an illegal value.

**NE\_ENUM\_INT\_2**

On entry, $\text{how\_many} = \langle value \rangle$, $n = \langle value \rangle$ and $\text{mm} = \langle value \rangle$.

Constraint: if $\text{how\_many} = \text{Nag\_ComputeAll}$, $\text{mm} \geq n$;

otherwise $\text{mm} \geq$ the number of selected eigenvalues.

On entry, $\text{job} = \langle value \rangle$, $\text{pdvl} = \langle value \rangle$, $\text{mm} = \langle value \rangle$.

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

On entry, $\text{job} = \langle value \rangle$, $\text{pdvl} = \langle value \rangle$ and $n = \langle value \rangle$.

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

On entry, $\text{job} = \langle value \rangle$, $\text{pdvr} = \langle value \rangle$, $\text{mm} = \langle value \rangle$.

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

On entry, $\text{job} = \langle value \rangle$, $\text{pdvr} = \langle value \rangle$ and $n = \langle value \rangle$.

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

**NE\_INT**

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

Constraint: $n > 0$.

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

Constraint: $n \geq 0$.

On entry, $\text{pda} = \langle value \rangle$.

Constraint: $\text{pda} > 0$.

On entry, $\text{pdb} = \langle value \rangle$.

Constraint: $\text{pdb} > 0$. 
On entry, \( pdvl = \langle \text{value} \rangle \).
Constraint: \( pdvl > 0 \).

On entry, \( pdvr = \langle \text{value} \rangle \).
Constraint: \( pdvr > 0 \).

**NE_INT_2**

On entry, \( pda = \langle \text{value} \rangle \) and \( n = \langle \text{value} \rangle \).
Constraint: \( pda \geq n \).

On entry, \( pdb = \langle \text{value} \rangle \) and \( n = \langle \text{value} \rangle \).
Constraint: \( pdb \geq n \).

**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 3.6.6 in the Essential Introduction for further information.

**NE_NO_LICENCE**

Your licence key may have expired or may not have been installed correctly. See Section 3.6.5 in the Essential Introduction for further information.

7  Accuracy

None.

8  Parallelism and Performance

\texttt{f08yyc} is not threaded by NAG in any implementation.

\texttt{f08yyc} 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

An approximate asymptotic error bound on the chordal distance between the computed eigenvalue \( \tilde{\lambda} \) and the corresponding exact eigenvalue \( \lambda \) is
\[
\chi(\tilde{\lambda}, \lambda) \leq \epsilon \|(A, B)\|_F / S(\lambda)
\]
where \( \epsilon \) is the \textit{machine precision}.

An approximate asymptotic error bound for the right or left computed eigenvectors \( \tilde{x} \) or \( \tilde{y} \) corresponding to the right and left eigenvectors \( x \) and \( y \) is given by
\[
\theta(\tilde{z}, z) \leq \epsilon \|(A, B)\|_F / \text{Dif}.
\]

The real analogue of this function is \texttt{f08yle}.
10 Example

This example estimates condition numbers and approximate error estimates for all the eigenvalues and right eigenvectors of the pair \((S, T)\) given by

\[
S = \begin{pmatrix}
4.0 + 4.0i & 1.0 + 1.0i & 1.0 + 1.0i & 2.0 - 1.0i \\
0 & 2.0 + 1.0i & 1.0 + 1.0i & 1.0 + 1.0i \\
0 & 0 & 2.0 - 1.0i & 1.0 + 1.0i \\
0 & 0 & 0 & 6.0 - 2.0i
\end{pmatrix}
\]

and

\[
T = \begin{pmatrix}
2.0 & 1.0 + 1.0i & 1.0 + 1.0i & 3.0 - 1.0i \\
0 & 1.0 & 2.0 + 1.0i & 1.0 + 1.0i \\
0 & 0 & 1.0 & 1.0 + 1.0i \\
0 & 0 & 0 & 2.0
\end{pmatrix}.
\]

The eigenvalues and eigenvectors are computed by calling nag_ztgevc (f08zxc).

10.1 Program Text

/* nag_ztgsna (f08zyc) Example Program. 
 * Copyright 2014 Numerical Algorithms Group. 
 * Mark 23, 2011. 
 */

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

int main(void)
{
    /* Scalars */
    double eps, snorm, stnrm, tnorm, tol;
    Integer i, j, m, n, pds, pdt, pdvl, pdvr;
    Integer exit_status = 0;

    /* Arrays */
    Complex *s = 0, *t = 0, *vl = 0, *vr = 0;
    double *dif = 0, *scon = 0;

    /* Nag Types */
    NagError fail;
    NagOrderType order;

    #ifdef NAG_COLUMN_MAJOR
    #define S(I, J) s[(J-1)*pds +I-1 ]
    #define T(I, J) t[(J-1)*pdt +I-1 ]
    #define Nag_ColMajor Nag_RowMajor
    #else
    #define S(I, J) s[(I-1)*pds +J-1 ]
    #define T(I, J) t[(I-1)*pdt +J-1 ]
    #define Nag_RowMajor Nag_ColMajor
    #endif

    INIT_FAIL(fail);

    printf("nag_ztgsna (f08zyc) Example Program Results\n\n");

    /* Skip heading in data file */
    #ifdef _WIN32
    scanf_s("%[^
");
    #else

Mark 25

f08yyc.7

f08yyc
```c
scanf("%*\n");
#endif
#ifdef _WIN32
    scanf_s("%"NAG_IFMT"%*\n");
#else
    scanf("%"NAG_IFMT"%*\n");
#endif
if (n < 0)
{
    printf("Invalid n\n");
    exit_status = 1;
    goto END;
}
m = n;
pds = n;
pdt = n;
pdvl = n;
pdvr = n;

/* Allocate memory */
if (!(dif = NAG_ALLOC(n, double)) ||
    !(scon = NAG_ALLOC(n, double)) ||
    !(s = NAG_ALLOC(n*n, Complex)) ||
    !(t = NAG_ALLOC(n*n, Complex)) ||
    !(vl = NAG_ALLOC(n*m, Complex)) ||
    !(vr = NAG_ALLOC(n*m, Complex)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Read S and T from data file */
for (i = 1; i <= n; ++i)
    for (j = 1; j <= n; ++j)
#ifndef _WIN32
    scanf_s(" ( %lf , %lf )", &S(i, j).re, &S(i, j).im);
#else
    scanf(" ( %lf , %lf )", &S(i, j).re, &S(i, j).im);
#endif
#ifndef _WIN32
    scanf("%*\n");
#else
    scanf("%*\n");
#endif
for (i = 1; i <= n; ++i)
    for (j = 1; j <= n; ++j)
#ifndef _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
#ifndef _WIN32
    scanf("%*\n");
#else
    scanf("%*\n");
#endif

/* Calculate the left and right generalized eigenvectors of the matrix pair (S,T) using nag_ztgevc (f08yxc). NULL may be passed here in place of the select array since all eigenvectors are requested. */
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_ztgevc (f08yxc)\n", fail.message);
    exit_status = 1;
    goto END;
}
```

---

The code snippet above is from the NAG Library Manual, specifically from the documentation for the function `nag_ztgevc`, which is used for performing the generalized eigenvalue problem for a complex matrix pair. The code demonstrates how to read from a file, allocate memory for the matrices, and then calculate the eigenvectors using the `nag_ztgevc` function.
/* Estimate condition numbers for all the generalized eigenvalues and right * eigenvectors of the pair (S,T) using nag_ztgsna (f08yyc). * NULL may be passed here in place of the select array since all * eigenvectors are requested. */

nag_ztgsna(order, Nag_DoBoth, Nag_ComputeAll, NULL, n, s, pds, t, pdt, vl, pdvl, vr, pdvr, scon, dif, n, &m, &fail);

if (fail.code != NE_NOERROR)
{
    printf("Error from nag_ztgsna (f08yyc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print condition numbers of eigenvalues and right eigenvectors */

printf("Condition numbers of eigenvalues (scon) and right eigenvectors "
      "(diff),\n");

for (i = 0; i < m; ++i)
    printf("%10.1e\n ", scon[i]);

for (i = 0; i < m; ++i)
    printf("%10.1e\n ", dif[i]);

/* Compute the norm of (S,T) using nag_zge_norm (f16uac). */

eps = nag_machine_precision;

nag_zge_norm(order, Nag_OneNorm, n, n, s, pds, &snorm, &fail);

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;
}

if (snorm == 0.0)
    stnrm = ABS(tnorm);
else if (tnorm == 0.0)
    stnrm = ABS(snorm);
else
    stnrm = ABS(tnorm)*(1.0+(snorm/tnorm)*(snorm/tnorm));

printf("\nApproximate error estimates for eigenvalues of (S,T)\n");

/* Calculate approximate error estimates */

tol = eps*stnrm;

printf("\nError estimates for eigenvalues (errval) and right eigenvectors"
       " (errvec),\n";

for (i = 0; i < m; ++i)
    printf("%10.1e\n ", tol/scon[i]);

for (i = 0; i < m; ++i)
    printf("%10.1e\n ", tol/dif[i]);

END:
NAG_FREE(dif);
NAG_FREE(scon);
NAG_FREE(s);
NAG_FREE(t);
NAG_FREE(vl);
NAG_FREE(vr);
return exit_status;

10.2 Program Data

nag_ztgsna (f08yyc) Example Program Data

4 : n

( 4.0, 4.0) ( 1.0, 1.0) ( 1.0, 1.0) ( 2.0,-1.0) : matrix S
( 0.0, 0.0) ( 2.0, 1.0) ( 1.0, 1.0) ( 1.0, 1.0)
( 0.0, 0.0) ( 0.0, 0.0) ( 2.0,-1.0) ( 1.0, 1.0)
( 0.0, 0.0) ( 0.0, 0.0) ( 0.0, 0.0) ( 6.0,-2.0)

( 2.0, 0.0) ( 1.0, 1.0) ( 1.0, 1.0) ( 3.0,-1.0) : matrix T
( 0.0, 0.0) ( 1.0, 0.0) ( 2.0, 1.0) ( 1.0, 1.0)
( 0.0, 0.0) ( 0.0, 0.0) ( 1.0, 0.0) ( 1.0, 1.0)
( 0.0, 0.0) ( 0.0, 0.0) ( 0.0, 0.0) ( 2.0, 0.0)

10.3 Program Results

nag_ztgsna (f08yyc) Example Program Results

Condition numbers of eigenvalues (scon) and right eigenvectors (diff),
scon: 1.0e+00 8.2e-01 7.2e-01 8.2e-01
diff: 3.2e-01 3.6e-01 5.5e-01 2.8e-01
Approximate error estimates for eigenvalues of (S,T)

Error estimates for eigenvalues (errval) and right eigenvectors (errvec),
errval: 1.5e-15 1.9e-15 2.1e-15 1.9e-15
ervec: 4.8e-15 4.3e-15 2.8e-15 5.5e-15