1 Purpose

nag_dtgsna (f08ylc) estimates condition numbers for specified eigenvalues and/or eigenvectors of a matrix pair in generalized real Schur form.

2 Specification

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

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

3 Description

nag_dtgsna (f08ylc) estimates condition numbers for specified eigenvalues and/or right eigenvectors of an \( n \times n \) matrix pair \((S, T)\) in real 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 real generalized Schur form if \(S\) is block upper triangular with 1 by 1 and 2 by 2 diagonal blocks and \(T\) is upper triangular as returned, for example, by nag_dgges (f08xac) or nag_dggesx (f08xbc), or nag_dhgeqz (f08xec) with \(\text{job} = \text{Nag}_\text{Schur}\). The diagonal elements, or blocks, 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

\[
\lambda_i = \frac{\alpha_i}{\beta_i},
\]

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 \((A, B)\)

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

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

\[
s(\lambda) = \frac{\left( |y^T A x|^2 + |y^T B x|^2 \right)^{1/2}}{|x^T x||y^T y|},
\]

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.

The definition of the reciprocal of the estimated condition number of the right eigenvector \(x\) and the left eigenvector \(y\) corresponding to the simple eigenvalue \(\lambda\) depends upon whether \(\lambda\) is a real eigenvalue, or one of a complex conjugate pair.
If the eigenvalue $\lambda$ is real and $U$ and $V$ are orthogonal transformations such that

$$U^T(A, B) = (S, T) = \left( \begin{array}{cc} \alpha & * \\ 0 & S_{22} \end{array} \right) \left( \begin{array}{cc} \beta & * \\ 0 & T_{22} \end{array} \right),$$

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

$$\text{Dif}(x) \equiv \text{Dif}(y) = \text{Dif}((\alpha, \beta), (S_{22}, T_{22})) = \sigma_{\text{min}}(Z),$$

where $\sigma_{\text{min}}(Z)$ denotes the smallest singular value of the $2(n-1)$ by $2(n-1)$ matrix

$$Z = \left( \begin{array}{cc} \alpha \otimes I & -1 \otimes S_{22} \\ \beta \otimes I & -1 \otimes T_{22} \end{array} \right)$$

and $\otimes$ is the Kronecker product.

If $\lambda$ is part of a complex conjugate pair and $U$ and $V$ are orthogonal transformations such that

$$U^T(A, B) = (S, T) = \left( \begin{array}{cc} S_{11} & * \\ 0 & S_{22} \end{array} \right) \left( \begin{array}{cc} T_{11} & * \\ 0 & T_{22} \end{array} \right),$$

where $S_{11}$ and $T_{11}$ are two by two matrices, $S_{22}$ and $T_{22}$ are $(n-2)$ by $(n-2)$ matrices, and $(S_{11}, T_{11})$ corresponds to the complex conjugate eigenvalue pair $\lambda, \overline{\lambda}$, then there exist unitary matrices $U_1$ and $V_1$ such that

$$U_1^H S_{11} V_1 = \left( \begin{array}{cc} s_{11} & s_{12} \\ 0 & s_{22} \end{array} \right) \quad \text{and} \quad T_1^H T_{11} V_1 = \left( \begin{array}{cc} t_{11} & t_{12} \\ 0 & t_{22} \end{array} \right).$$

The eigenvalues are given by $\lambda = s_{11}/t_{11}$ and $\overline{\lambda} = s_{22}/t_{22}$. Then the Frobenius norm-based, estimated reciprocal condition number is bounded by

$$\text{Dif}(x) \equiv \text{Dif}(y) \leq \min(d_1, \max(1, |\text{Re}(s_{11})/\text{Re}(s_{22})|), d_2)$$

where $\text{Re}(z)$ denotes the real part of $z$, $d_1 = \text{Dif}((s_{11}, t_{11}), (s_{22}, t_{22})) = \sigma_{\text{min}}(Z_1)$, $Z_1$ is the complex two by two matrix

$$Z_1 = \left( \begin{array}{cc} s_{11} & -s_{22} \\ t_{11} & -t_{22} \end{array} \right),$$

and $d_2$ is an upper bound on $\text{Dif}((S_{11}, T_{11}), (S_{22}, T_{22}))$; i.e., an upper bound on $\sigma_{\text{min}}(Z_2)$, where $Z_2$ is the $(2n-2)$ by $(2n-2)$ matrix

$$Z_2 = \left( \begin{array}{cc} S_{11}^T \otimes I & -I \otimes S_{22} \\ T_{11}^T \otimes I & -I \otimes T_{22} \end{array} \right).$$

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: order $-$ Nag_OrderType

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 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

**Constraint:** order = Nag_RowMajor or Nag_ColMajor.

2: job – Nag_JobType  
*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: howmny – Nag_HowManyType  
*On entry:* indicates how many condition numbers are to be computed.

howmny = Nag_ComputeAll  
Condition numbers for all eigenpairs are computed.

howmny = Nag_ComputeSelected  
Condition numbers for selected eigenpairs (as specified by select) are computed.

**Constraint:** howmny = 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 howmny = Nag_ComputeSelected;

otherwise select may be NULL.

*On entry:* specifies the eigenpairs for which condition numbers are to be computed if howmny = 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 howmny = Nag_ComputeAll, select is not referenced and may be NULL.

5: n – Integer  
*Input*

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

**Constraint:** n ≥ 0.

6: a[dim] – const double  
*Input*

**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

- \(a[(j - 1) \times pda + i - 1]\) when order = Nag_ColMajor;
- \(a[(i - 1) \times pda + j - 1]\) when order = Nag_RowMajor.

*On entry:* the upper quasi-triangular matrix \(S\).
7: \( \text{pda} \) – Integer \hspace{1cm} \text{Input} \\
On \ entry: \ the \ stride \ separating \ row \ or \ column \ elements \ (depending \ on \ the \ value \ of \ \text{order}) \ in \ the \ array \ \text{a}. \\
Constraint: \ \text{pda} \geq \max(1, \text{n}).

8: \( \text{b}[\text{dim}] \) – const double \hspace{1cm} \text{Input} \\
Note: \ the \ dimension, \ \text{dim}, \ of \ the \ array \ \text{b} \ must \ be \ at \ least \ \text{pdb} \times \text{n}.

The \( (i, j) \)th \ element \ of \ the \ matrix \ \text{B} \ is \ stored \ in \\
\begin{align*}
  \text{b}[(j-1) \times \text{pdb} + i - 1] & \quad \text{when} \ \text{order} = \text{Nag\_ColMajor}; \\
  \text{b}[(i-1) \times \text{pdb} + j - 1] & \quad \text{when} \ \text{order} = \text{Nag\_RowMajor}.
\end{align*}

On \ entry: \ the \ upper \ triangular \ matrix \ \text{T}.

9: \( \text{pdb} \) – Integer \hspace{1cm} \text{Input} \\
On \ entry: \ the \ stride \ separating \ row \ or \ column \ elements \ (depending \ on \ the \ value \ of \ \text{order}) \ in \ the \ array \ \text{b}. \\
Constraint: \ \text{pdb} \geq \max(1, \text{n}).

10: \( \text{vl}[\text{dim}] \) – const double \hspace{1cm} \text{Input} \\
Note: \ the \ dimension, \ \text{dim}, \ of \ the \ array \ \text{vl} \ must \ be \ at \ least \\
\begin{align*}
  \text{pdvl} \times \text{mm} & \quad \text{when} \ \text{job} = \text{Nag\_EigVals} \ or \ \text{Nag\_DoBoth} \ and \ \text{order} = \text{Nag\_ColMajor}; \\
  \text{n} \times \text{pdvl} & \quad \text{when} \ \text{job} = \text{Nag\_EigVals} \ or \ \text{Nag\_DoBoth} \ and \ \text{order} = \text{Nag\_RowMajor}; \\
  \text{vl} & \quad \text{otherwise} \ \text{vl} \ \text{may \ be} \ \text{NULL}.
\end{align*}

The \( (i, j) \)th \ element \ of \ the \ matrix \ is \ stored \ in \\
\begin{align*}
  \text{vl}[(j-1) \times \text{pdvl} + i - 1] & \quad \text{when} \ \text{order} = \text{Nag\_ColMajor}; \\
  \text{vl}[(i-1) \times \text{pdvl} + j - 1] & \quad \text{when} \ \text{order} = \text{Nag\_RowMajor}.
\end{align*}

On \ entry: \ if \ \text{job} = \text{Nag\_EigVals} \ or \ \text{Nag\_DoBoth}, \ \text{vl} \ must \ contain \ left \ eigenvectors \ of \ (\text{S}, \text{T}), \\
\begin{align*}
  \text{corresponding \ to \ the \ eigenpairs \ specified \ by} \ \text{howmny} \ \text{and} \ \text{select}. \ \text{The} \ \text{eigenvectors} \ \text{must \ be \ stored} \\
  \text{in \ consecutive \ columns \ of} \ \text{vl}, \ \text{as \ returned \ by} \ \text{nag\_dggev} (f08wac) \ \text{or} \ \text{nag\_dtgevc} (f08ykc).
\end{align*}

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

11: \( \text{pdvl} \) – Integer \hspace{1cm} \text{Input} \\
On \ entry: \ the \ stride \ separating \ row \ or \ column \ elements \ (depending \ on \ the \ value \ of \ \text{order}) \ in \ the \ array \ \text{vl}. \\
Constraints:
\begin{align*}
  \text{if} \ \text{order} = \text{Nag\_ColMajor}, \\
  \text{if} \ \text{job} = \text{Nag\_EigVals} \ or \ \text{Nag\_DoBoth}, \ \text{pdvl} \geq \text{n}; \\
  \text{otherwise} \ \text{pdvl} \geq 1.; \\
  \text{if} \ \text{order} = \text{Nag\_RowMajor}, \\
  \text{if} \ \text{job} = \text{Nag\_EigVals} \ or \ \text{Nag\_DoBoth}, \ \text{pdvl} \geq \text{mm}; \\
  \text{otherwise} \ \text{vl} \ \text{may \ be} \ \text{NULL}.
\end{align*}

12: \( \text{vr}[\text{dim}] \) – const double \hspace{1cm} \text{Input} \\
Note: \ the \ dimension, \ \text{dim}, \ of \ the \ array \ \text{vr} \ must \ be \ at \ least \\
\begin{align*}
  \text{pdvr} \times \text{mm} & \quad \text{when} \ \text{job} = \text{Nag\_EigVals} \ or \ \text{Nag\_DoBoth} \ and \ \text{order} = \text{Nag\_ColMajor}; \\
  \text{n} \times \text{pdvr} & \quad \text{when} \ \text{job} = \text{Nag\_EigVals} \ or \ \text{Nag\_DoBoth} \ and \ \text{order} = \text{Nag\_RowMajor}; \\
  \text{vr} & \quad \text{otherwise} \ \text{vr} \ \text{may \ be} \ \text{NULL}.
\end{align*}
The \((i,j)\)th element of the matrix is stored in
\[ vr[(j-1) \times pdvr + i - 1] \text{ when } order = \text{Nag\_ColMajor}; \]
\[ vr[(i-1) \times pdvr + j - 1] \text{ when } order = \text{Nag\_RowMajor}. \]

On entry: if \(job = \text{Nag\_EigVals or Nag\_DoBoth}, vr\) must contain right eigenvectors of \((S,T)\), corresponding to the eigenpairs specified by \text{howmny} and \text{select}. The eigenvectors must be stored in consecutive columns of \(vr\), as returned by \text{nag\_dggev (f08wac)} or \text{nag\_dtgevc (f08ykc)}.

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

13: \(pdvr\) – Integer

\text{Input}

On entry: the stride separating row or column elements (depending on the value of \text{order}) in the array \(vr\).

Constraints:

if \(order = \text{Nag\_ColMajor},\)

if \(job = \text{Nag\_EigVals or Nag\_DoBoth}, pdvr \geq n;\)

otherwise \(pdvr \geq 1.;\)

if \(order = \text{Nag\_RowMajor},\)

if \(job = \text{Nag\_EigVals or Nag\_DoBoth}, pdvr \geq mm;\)

otherwise \(vr\) may be \text{NULL}.

14: \(s[\text{dim}]\) – double

\text{Output}

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

\(\text{mm}\) when \(job = \text{Nag\_EigVals or Nag\_DoBoth};\)

otherwise \(s\) may be \text{NULL}.

On exit: if \(job = \text{Nag\_EigVals or Nag\_DoBoth}, the reciprocal condition numbers of the selected eigenvalues, stored in consecutive elements of the array. For a complex conjugate pair of eigenvalues two consecutive elements of \(s\) are set to the same value. Thus \(s[j - 1], \text{dif}[j - 1]\), and the \(j\)th columns of \(VL\) and \(VR\) all correspond to the same eigenpair (but not in general the \(j\)th eigenpair, unless all eigenpairs are selected).

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

15: \(\text{dif}[\text{dim}]\) – double

\text{Output}

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

\(\text{mm}\) when \(job = \text{Nag\_EigVecs or Nag\_DoBoth};\)

otherwise \(\text{dif}\) may be \text{NULL}.

On exit: if \(job = \text{Nag\_EigVecs or Nag\_DoBoth}, the estimated reciprocal condition numbers of the selected eigenvectors, stored in consecutive elements of the array. For a complex eigenvector two consecutive elements of \(\text{dif}\) are set to the same value. 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 \(job = \text{Nag\_EigVals}, \text{dif}\) is not referenced and may be \text{NULL}.

16: \(\text{mm}\) – Integer

\text{Input}

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

Constraints:

if \(\text{howmny} = \text{Nag\_ComputeAll}, \text{mm} \geq n;\)

otherwise \(\text{mm} \geq m.\)
17: \textbf{m} – Integer * \hspace{1cm} \textit{Output}

\textit{On exit:} the number of elements of the arrays \textbf{s} and \textbf{dif} used to store the specified condition numbers; for each selected real eigenvalue one element is used, and for each selected complex conjugate pair of eigenvalues, two elements are used. If \textbf{howmny} = Nag_ComputeAll, \textbf{m} is set to \textbf{n}.

18: \textbf{fail} – NagError * \hspace{1cm} \textit{Input/Output}

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

6 \hspace{1cm} \textbf{Error Indicators and Warnings}

\textbf{NE_ALLOC_FAIL}

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

\textbf{NE_BAD_PARAM}

On entry, argument \langle\textit{value}\rangle had an illegal value.

\textbf{NE_ENUM_INT_2}

On entry, \textbf{job} = \langle\textit{value}\rangle, \textbf{pdvl} = \langle\textit{value}\rangle, \textbf{mm} = \langle\textit{value}\rangle.
Constraint: if \textbf{job} = Nag_EigVals or Nag_DoBoth, \textbf{pdvl} \geq \textbf{mm}.

On entry, \textbf{job} = \langle\textit{value}\rangle, \textbf{pdvl} = \langle\textit{value}\rangle and \textbf{n} = \langle\textit{value}\rangle.
Constraint: if \textbf{job} = Nag_EigVals or Nag_DoBoth, \textbf{pdvl} \geq \textbf{n}.

On entry, \textbf{job} = \langle\textit{value}\rangle, \textbf{pdvr} = \langle\textit{value}\rangle, \textbf{mm} = \langle\textit{value}\rangle.
Constraint: if \textbf{job} = Nag_EigVals or Nag_DoBoth, \textbf{pdvr} \geq \textbf{mm}.

On entry, \textbf{job} = \langle\textit{value}\rangle, \textbf{pdvr} = \langle\textit{value}\rangle and \textbf{n} = \langle\textit{value}\rangle.
Constraint: if \textbf{job} = Nag_EigVals or Nag_DoBoth, \textbf{pdvr} \geq \textbf{n}.

\textbf{NE_ENUM_INT_3}

On entry, \textbf{howmny} = \langle\textit{value}\rangle, \textbf{n} = \langle\textit{value}\rangle, \textbf{mm} = \langle\textit{value}\rangle and \textbf{m} = \langle\textit{value}\rangle.
Constraint: if \textbf{howmny} = Nag_ComputeAll, \textbf{mm} \geq \textbf{n}; otherwise \textbf{mm} \geq \textbf{m}.

\textbf{NE_INT}

On entry, \textbf{n} = \langle\textit{value}\rangle.
Constraint: \textbf{n} \geq 0.

On entry, \textbf{pda} = \langle\textit{value}\rangle.
Constraint: \textbf{pda} > 0.

On entry, \textbf{pdb} = \langle\textit{value}\rangle.
Constraint: \textbf{pdb} > 0.

On entry, \textbf{pdvl} = \langle\textit{value}\rangle.
Constraint: \textbf{pdvl} > 0.

On entry, \textbf{pdvr} = \langle\textit{value}\rangle.
Constraint: \textbf{pdvr} > 0.

\textbf{NE_INT_2}

On entry, \textbf{pda} = \langle\textit{value}\rangle, and \textbf{n} = \langle\textit{value}\rangle.
Constraint: \textbf{pda} \geq \text{max}(1, \textbf{n}).

On entry, \textbf{pdb} = \langle\textit{value}\rangle, and \textbf{n} = \langle\textit{value}\rangle.
Constraint: \textbf{pdb} \geq \text{max}(1, \textbf{n}).
7 Accuracy

None.

8 Parallelism and Performance

nag_dtgsna (f08ylc) is not threaded by NAG in any implementation.

nag_dtgsna (f08ylc) 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 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

\[ \theta(\tilde{x}, x) \leq \epsilon \| (A, B) \|_F / \text{Dif}. \]

The complex analogue of this function is nag_ztgsna (f08yyc).

10 Example

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

\[ S = \begin{pmatrix} 4.0 & 1.0 & 1.0 & 2.0 \\ 0 & 3.0 & -1.0 & 1.0 \\ 0 & 1.0 & 3.0 & 1.0 \\ 0 & 0 & 0 & 6.0 \end{pmatrix} \quad \text{and} \quad T = \begin{pmatrix} 2.0 & 1.0 & 1.0 & 3.0 \\ 0 & 1.0 & 0.0 & 1.0 \\ 0 & 0 & 1.0 & 1.0 \\ 0 & 0 & 0 & 2.0 \end{pmatrix}. \]

The eigenvalues and eigenvectors are computed by calling nag_dtgevc (f08ykc).
10.1 Program Text

/* nag_dtgsna (f08ylc) Example Program. */
* Copyright 2014 Numerical Algorithms Group.
* Mark 23, 2011. */

#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <naq_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 */
    double *dif = 0, *s = 0, *scon = 0, *t = 0, *vl = 0, *vr = 0;

    /* Nag Types */
    NagError fail;
    Nag_OrderType 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]
    order = Nag_ColMajor;
    #else
    #define S(I, J) s[(I-1)*pds+J-1]
    #define T(I, J) t[(I-1)*pdt +J-1]
    order = Nag_RowMajor;
    #endif

    INIT_FAIL(fail);

    printf("nag_dtgsna (f08ylc) 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

    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, double)) ||
!(t = NAG_ALLOC(n*n, double)) ||
!(vl = NAG_ALLOC(n*m, double)) ||
!(vr = NAG_ALLOC(n*m, double)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}
/* Read S and T from data file */
for (i = 1; i <= n; ++i)
#if defined _WIN32
    for (j = 1; j <= n; ++j) scanf_s("%lf", &S(i, j));
#else
    for (j = 1; j <= n; ++j) scanf("%lf", &S(i, j));
#endif
#if defined _WIN32
    scanf_s("%*\n");
#else
    scanf("%*\n");
#endif
for (i = 1; i <= n; ++i)
#if defined _WIN32
    for (j = 1; j <= n; ++j) scanf_s("%lf", &T(i, j));
#else
    for (j = 1; j <= n; ++j) scanf("%lf", &T(i, j));
#endif
#if defined _WIN32
    scanf_s("%*\n");
#else
    scanf("%*\n");
#endif
/* Calculate the left and right generalized eigenvectors of the matrix pair (S,T) using nag_dtgevc (f08ykc).
 * NULL may be passed here in place of the select array since all eigenvectors are requested.
 */
nag_dtgevc(order, Nag_BothSides, Nag_ComputeAll, NULL, n, s, pds, t, pdt,
            vl, pdvl, vr, pdvr, n, &m, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dtgevc (f08ykc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Estimate condition numbers for all the generalized eigenvalues and right eigenvectors of the pair (S,T) using nag_dtgsna (f08ylc).
 * NULL may be passed here in place of the select array since all eigenvectors are requested.
 */
nag_dtgsna(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_dtgsna (f08ylc).\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
"(\n"
"diff),\n"
"\n")
printf("scon: ");
for (i = 0; i < m; ++i)
    printf("%10.1e%s", scon[i], i%7 == 6? \n "":"");
printf("ndif: ");
for (i = 0; i < m; ++i)
    printf("%10.1e%s", dif[i], i%7 == 6? \n "":"");
/* Compute the norm of (S,T) using nag_dge_norm (f16rac). */

eps = nag_machine_precision;
nag_dge_norm(order, Nag_OneNorm, n, n, s, pds, &snorm, &fail);
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", fail.message);
    exit_status = 1;
    goto END;
}
if (snorm == 0.0)
    stnrm = ABS(tnorm);
else if (tnorm == 0.0)
    stnrm = ABS(snorm);
else if (ABS(snorm) >= ABS(tnorm))
    stnrm = ABS(tnorm)*sqrt(1.0+(tnorm/snorm)*(tnorm/snorm));
else
    stnrm = ABS(tnorm)*sqrt(1.0+(snorm/tnorm)*(snorm/tnorm));

/* Calculate approximate error estimates */
tol = eps*stnrm;

printf("\nError estimates for eigenvalues (errval) and right eigenvectors" "\n(errvec),\n");
printf("errval: ");
for (i = 0; i < m; ++i)
    printf(" %10.1e%s", tol/scon[i], i%7 == 6?"\n":"");
printf("\n(errvec: ");
for (i = 0; i < m; ++i)
    printf(" %10.1e%s", tol/dif[i], i%7 == 6?"\n":"");

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_dtgsna (f08ylc) Example Program Data

4 : n

4.0 1.0 1.0 2.0
0.0 3.0 -1.0 1.0
0.0 1.0 3.0 1.0
0.0 0.0 0.0 6.0 : matrix S

2.0 1.0 1.0 3.0
0.0 1.0 0.0 1.0
0.0 0.0 1.0 1.0
0.0 0.0 0.0 2.0 : matrix T
10.3 Program Results

nag_dtsna (f08yclc) Example Program Results

Condition numbers of eigenvalues (scon) and right eigenvectors (dif),
\begin{align*}
\text{scon:} & \quad 1.6e+00 \quad 1.7e+00 \quad 1.7e+00 \quad 1.4e+00 \\
\text{dif:} & \quad 5.4e-01 \quad 1.5e-01 \quad 1.5e-01 \quad 1.2e-01
\end{align*}

Error estimates for eigenvalues (errval) and right eigenvectors (errvec),
\begin{align*}
\text{errval:} & \quad 8.7e-16 \quad 7.8e-16 \quad 7.8e-16 \quad 9.9e-16 \\
\text{errvec:} & \quad 2.5e-15 \quad 9.0e-15 \quad 9.0e-15 \quad 1.1e-14
\end{align*}