NAG C Library Function Document

nag_dtrevc (f08qkc)

1 Purpose

nag_dtrevc (f08qkc) computes selected left and/or right eigenvectors of a real upper quasi-triangular matrix.

2 Specification

```
void nag_dtrevc (Nag_OrderType order, Nag_SideType side, Nag_HowManyType how_many,
     Boolean select[], Integer n, const double t[], Integer pdt, double vl[],
     Integer pdvl, double vr[], Integer pdvr, Integer mm, Integer *m,
     NagError *fail)
```

3 Description

nag_dtrevc (f08qkc) computes left and/or right eigenvectors of a real upper quasi-triangular matrix T in canonical Schur form. Such a matrix arises from the Schur factorization of a real general matrix, as computed by nag_dhseqr (f08pec), for example.

The right eigenvector x, and the left eigenvector y, corresponding to an eigenvalue λ, are defined by:

$$Tx = \lambda x \quad \text{and} \quad y^H T = \lambda y^H \text{ (or } T^T y = \bar{\lambda} y).$$

Note that even though T is real, λ, x and y may be complex. If x is an eigenvector corresponding to a complex eigenvalue λ, then the complex conjugate vector $\bar{x}$ is the eigenvector corresponding to the complex conjugate eigenvalue $\bar{\lambda}$.

The function can compute the eigenvectors corresponding to selected eigenvalues, or it can compute all the eigenvectors. In the latter case the eigenvectors may optionally be pre-multiplied by an input matrix Q. Normally Q is an orthogonal matrix from the Schur factorization of a matrix A as $A = QTQ^T$; if x is a (left or right) eigenvector of T, then Qx is an eigenvector of A.

The eigenvectors are computed by forward or backward substitution. They are scaled so that, for a real eigenvector x, $\max(|x_i|) = 1$, and for a complex eigenvector, $\max(|\mathrm{Re}(x_i)| + |\mathrm{Im}(x_i)|) = 1$.

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: **side** – Nag_SideType *Input*

On entry: indicates whether left and/or right eigenvectors are to be computed as follows:

if **side** = **Nag_RightSide**, only right eigenvectors are computed;

if **side** = **Nag_LeftSide**, only left eigenvectors are computed;

if **side** = **Nag_BothSides**, both left and right eigenvectors are computed.

Constraint: **side** = **Nag_RightSide**, **Nag_LeftSide** or **Nag_BothSides**.

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

On entry: indicates how many eigenvectors are to be computed as follows:

if **how_many** = **Nag_ComputeAll**, all eigenvectors (as specified by **side**) are computed;

if **how_many** = **Nag_BackTransform**, all eigenvectors (as specified by **side**) are computed and then pre-multiplied by the matrix Q (which is overwritten);

if **how_many** = **Nag_ComputeSelected**, selected eigenvectors (as specified by **side** and **select**) are computed.

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

4: **select**[dim] – Boolean *Input/Output*

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 which eigenvectors are to be computed if **how_many** = **Nag_ComputeSelected**. To obtain the real eigenvector corresponding to the real eigenvalue λ_j, **select**[j] must be set **TRUE**. To select the complex eigenvector corresponding to a complex conjugate pair of eigenvalues λ_j and λ_{j+1}, **select**[j] and/or **select**[$j+1$] must be set **TRUE**; the eigenvector corresponding to the **first** eigenvalue in the pair is computed.

On exit: if a complex eigenvector was selected as specified above, then **select**[j] is set to **TRUE** and **select**[$j+1$] to **FALSE**.

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

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

8: **vl**[dim] – double *Input/Output*

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

$\max(1, \mathbf{pdvl} \times \mathbf{mm})$ when **side** = **Nag_LeftSide** or **Nag_BothSides** and **order** = **Nag_ColMajor**;

$\max(1, \mathbf{pdvl} \times \mathbf{n})$ when **side** = **Nag_LeftSide** or **Nag_BothSides** and **order** = **Nag_RowMajor**;

1 when **side** = **Nag_RightSide**.

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 **how_many** = **Nag_BackTransform** and **side** = **Nag_LeftSide** or **Nag_BothSides**, **vl** must contain an n by n matrix Q (usually the matrix of Schur vectors returned by nag_dhseqr (f08pec)). If **how_many** = **Nag_ComputeAll** or **Nag_ComputeSelected**, **vl** need not be set.

On exit: if **side** = **Nag_LeftSide** or **Nag_BothSides**, **vl** contains the computed left eigenvectors (as specified by **how_many** and **select**). The eigenvectors are stored consecutively in the rows or columns of the array, in the same order as their eigenvalues. Corresponding to each real eigenvalue is a real eigenvector, occupying one row or column. Corresponding to each complex conjugate pair of eigenvalues, is a complex eigenvector occupying two rows or columns; the first row or column holds the real part and the second row or column holds the imaginary part.

vl is not referenced if **side** = **Nag_RightSide**.

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 **side** = **Nag_LeftSide** or **Nag_BothSides**, **pdvl** $\geq \max(1, \mathbf{n})$;
  if **side** = **Nag_RightSide**, **pdvl** ≥ 1;

if **order** = **Nag_RowMajor**,
  if **side** = **Nag_LeftSide** or **Nag_BothSides**, **pdvl** $\geq \max(1, \mathbf{mm})$;
  if **side** = **Nag_RightSide**, **pdvl** ≥ 1.

10: **vr**[dim] – double *Input/Output*

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

$\max(1, \mathbf{pdvr} \times \mathbf{mm})$ when **side** = **Nag_RightSide** or **Nag_BothSides** and **order** = **Nag_ColMajor**;

$\max(1, \mathbf{pdvr} \times \mathbf{n})$ when **side** = **Nag_RightSide** or **Nag_BothSides** and **order** = **Nag_RowMajor**;

1 when **side** = **Nag_LeftSide**.

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

On entry: if **how_many** = **Nag_BackTransform** and **side** = **Nag_RightSide** or **Nag_BothSides**, **vr** must contain an n by n matrix Q (usually the matrix of Schur vectors returned by nag_dhseqr (f08pec)). If **how_many** = **Nag_ComputeAll** or **Nag_ComputeSelected**, **vr** need not be set.

On exit: if **side** = **Nag_RightSide** or **Nag_BothSides**, **vr** contains the computed right eigenvectors (as specified by **how_many** and **select**). The eigenvectors are stored consecutively in the rows or columns of the array, in the same order as their eigenvalues. Corresponding to each real eigenvalue is a real eigenvector, occupying one row or column. Corresponding to each complex conjugate pair of eigenvalues, is a complex eigenvector occupying two rows or columns; the first row or column holds the real part and the second row or column holds the imaginary part.

vr is not referenced if **side** = **Nag_LeftSide**.

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 **side** = **Nag_RightSide** or **Nag_BothSides**, **pdvr** $\geq \max(1, \mathbf{n})$;
  if **side** = **Nag_LeftSide**, **pdvr** ≥ 1;

if **order** = **Nag_RowMajor**,
  if **side** = **Nag_RightSide** or **Nag_BothSides**, **pdvr** $\geq \max(1, \mathbf{mm})$;

if **side** = **Nag_LeftSide**, **pdvr** ≥ 1.

12: **mm** – Integer *Input*

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

Constraint: **mm** $\geq required\_rowcol$.

13: **m** – Integer * *Output*

On exit: $required\_rowcol$, the number of rows or columns of **vl** and/or **vr** actually used to store the computed eigenvectors. If **how_many** = **Nag_ComputeAll** or **Nag_BackTransform**, **m** is set to n.

14: **fail** – NagError * *Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 0.

On entry, **mm** = ⟨*value*⟩.
Constraint: **mm** $\geq required\_rowcol$, where $required\_rowcol$ is obtained by counting 1 for each selected real eigenvector and 2 for each selected complex eigenvector.

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

NE_ENUM_INT_2

On entry, **side** = ⟨*value*⟩, **n** = ⟨*value*⟩, **pdvl** = ⟨*value*⟩.
Constraint: if **side** = **Nag_LeftSide** or **Nag_BothSides**, **pdvl** $\geq \max(1, \mathbf{n})$;
if **side** = **Nag_RightSide**, **pdvl** ≥ 1.

On entry, **side** = ⟨*value*⟩, **n** = ⟨*value*⟩, **pdvr** = ⟨*value*⟩.
Constraint: if **side** = **Nag_RightSide** or **Nag_BothSides**, **pdvr** $\geq \max(1, \mathbf{n})$;
if **side** = **Nag_LeftSide**, **pdvr** ≥ 1.

On entry, **side** = ⟨*value*⟩, **mm** = ⟨*value*⟩, **pdvl** = ⟨*value*⟩.
Constraint: if **side** = **Nag_LeftSide** or **Nag_BothSides**, **pdvl** $\geq \max(1, \mathbf{mm})$;
if **side** = **Nag_RightSide**, **pdvl** ≥ 1.

On entry, **side** = ⟨*value*⟩, **mm** = ⟨*value*⟩, **pdvr** = ⟨*value*⟩.
Constraint: if **side** = **Nag_RightSide** or **Nag_BothSides**, **pdvr** $\geq \max(1, \mathbf{mm})$;
if **side** = **Nag_LeftSide**, **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

If x_i is an exact right eigenvector, and $\tilde{x}_i$ is the corresponding computed eigenvector, then the angle $\theta(\tilde{x}_i, x_i)$ between them is bounded as follows:

$$\theta(\tilde{x}_i, x_i) \leq \frac{c(n)\epsilon\|T\|_2}{sep_i}$$

where sep_i is the reciprocal condition number of x_i.

The condition number sep_i may be computed by calling nag_dtrsna (f08qlc).

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_ztrevc (f08qxc).

9 Example

See Section 9 of the document for nag_dgebal (f08nhc).