

# NAG 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

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

void nag_dtrevc (Nag_OrderType order, Nag_SideType side,
                 Nag_HowManyType how_many, Nag_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  $\lambda$ , are defined by:

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

Note that even though  $T$  is real,  $\lambda$ ,  $x$  and  $y$  may be complex. If  $x$  is an eigenvector corresponding to a complex eigenvalue  $\lambda$ , 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,  $\max(|\text{Re}(x_i)| + |\text{Im}(x_i)|) = 1$ .

### 4 References

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

### 5 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: **side** – Nag\_SideType *Input*  
*On entry:* indicates whether left and/or right eigenvectors are to be computed.  
**side** = Nag\_RightSide  
 Only right eigenvectors are computed.  
**side** = Nag\_LeftSide  
 Only left eigenvectors are computed.  
**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.  
**how\_many** = Nag\_ComputeAll  
 All eigenvectors (as specified by **side**) are computed.  
**how\_many** = Nag\_BackTransform  
 All eigenvectors (as specified by **side**) are computed and then pre-multiplied by the matrix  $Q$  (which is overwritten).  
**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*] – Nag\_Boolean *Input/Output*  
**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 which eigenvectors are to be computed if **how\_many** = Nag\_ComputeSelected. To obtain the real eigenvector corresponding to the real eigenvalue  $\lambda_j$ , **select**[ $j-1$ ] must be set Nag\_TRUE. To select the complex eigenvector corresponding to a complex conjugate pair of eigenvalues  $\lambda_j$  and  $\lambda_{j+1}$ , **select**[ $j-1$ ] and/or **select**[ $j$ ] must be set Nag\_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**[ $j-1$ ] is set to Nag\_TRUE and **select**[ $j$ ] to Nag\_FALSE.  
 If **how\_many** = Nag\_ComputeAll or Nag\_BackTransform, **select** is not referenced and may be **NULL**.
- 5: **n** – Integer *Input*  
*On entry:*  $n$ , the order of the matrix  $T$ .  
*Constraint:*  $n \geq 0$ .
- 6: **t**[*dim*] – const double *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 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*] – double *Input/Output*  
**Note:** the dimension, *dim*, of the array **vl** must be at least  
**pdrv**  $\times$  **mm** when **side** = Nag\_LeftSide or Nag\_BothSides and **order** = Nag\_ColMajor;  
**n**  $\times$  **pdrv** when **side** = Nag\_LeftSide or Nag\_BothSides and **order** = Nag\_RowMajor;  
otherwise **vl** may be **NULL**.  
The (*i*, *j*)th element of the matrix is stored in  
**vl**[(*j* – 1)  $\times$  **pdrv** + *i* – 1] when **order** = Nag\_ColMajor;  
**vl**[(*i* – 1)  $\times$  **pdrv** + *j* – 1] when **order** = Nag\_RowMajor.  
*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.  
If **side** = Nag\_RightSide, **vl** is not referenced and may be **NULL**.
- 9: **pdrv** – 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 **side** = Nag\_LeftSide or Nag\_BothSides, **pdrv**  $\geq n$ ;  
if **side** = Nag\_RightSide, **vl** may be **NULL**.;  
if **order** = Nag\_RowMajor,  
if **side** = Nag\_LeftSide or Nag\_BothSides, **pdrv**  $\geq mm$ ;  
if **side** = Nag\_RightSide, **vl** may be **NULL**.
- 10: **vr**[*dim*] – double *Input/Output*  
**Note:** the dimension, *dim*, of the array **vr** must be at least  
**pdrv**  $\times$  **mm** when **side** = Nag\_RightSide or Nag\_BothSides and **order** = Nag\_ColMajor;  
**n**  $\times$  **pdrv** when **side** = Nag\_RightSide or Nag\_BothSides and **order** = Nag\_RowMajor;  
otherwise **vr** may be **NULL**.  
The (*i*, *j*)th element of the matrix is stored in  
**vr**[(*j* – 1)  $\times$  **pdrv** + *i* – 1] when **order** = Nag\_ColMajor;  
**vr**[(*i* – 1)  $\times$  **pdrv** + *j* – 1] when **order** = Nag\_RowMajor.  
*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.

If **side** = Nag\_LeftSide, **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 **side** = Nag\_RightSide or Nag\_BothSides, **pdvr**  $\geq$  **n**;  
     if **side** = Nag\_LeftSide, **vr** may be **NULL**;  
 if **order** = Nag\_RowMajor,  
     if **side** = Nag\_RightSide or Nag\_BothSides, **pdvr**  $\geq$  **mm**;  
     if **side** = Nag\_LeftSide, **vr** may be **NULL**.

- 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<sub>rowcol</sub>*, is **n** if **how\_many** = Nag\_ComputeAll or Nag\_BackTransform; if **how\_many** = Nag\_ComputeSelected, *required<sub>rowcol</sub>* is obtained by counting 1 for each selected real eigenvector and 2 for each selected complex eigenvector (see **select**), in which case  $0 \leq \text{required}_{\text{rowcol}} \leq n$ .

Constraints:

if **how\_many** = Nag\_ComputeAll or Nag\_BackTransform, **mm**  $\geq$  **n**;  
 otherwise **mm**  $\geq$  *required<sub>rowcol</sub>*.

- 13: **m** – Integer \* *Output*

On exit: *required<sub>rowcol</sub>*, 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 \* *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** =  $\langle value \rangle$ , **mm** =  $\langle value \rangle$  and **n** =  $\langle value \rangle$ .

Constraint: if **how\_many** = Nag\_ComputeAll or Nag\_BackTransform, **mm**  $\geq$  **n**;  
otherwise **mm**  $\geq$  *required\_rowcol*.

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

Constraint: if **side** = Nag\_LeftSide or Nag\_BothSides, **pdvl**  $\geq$  **mm**.

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

Constraint: if **side** = Nag\_LeftSide or Nag\_BothSides, **pdvl**  $\geq$  **n**.

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

Constraint: if **side** = Nag\_RightSide or Nag\_BothSides, **pdvr**  $\geq$  **mm**.

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

Constraint: if **side** = Nag\_RightSide or Nag\_BothSides, **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, **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**

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 Parallelism and Performance

nag\_dtrevc (f08qkc) 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\_ztrevc (f08qxc).

## 10 Example

See Section 10 in nag\_dgebal (f08nhc).

---