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

```c
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 \( \lambda \), are defined by:

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

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 \( \overline{x} \) 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


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_BothSide, both left and right eigenvectors are computed;

   Note: if how_many = Nag_Complete, then side is not referenced.

3: how_many – Nag_HowManyType

   Input

   On entry: indicates how many eigenvectors are required:

   if how_many = Nag_Complete, all eigenvectors are computed;
   if how_many = Nag_Selected, only the eigenvectors corresponding to the specified eigenvalues are computed.

4: select – Boolean

   Input

   On entry: specifies which eigenvectors are to be computed. Select[i] = true selects the \( i \)th eigenvector to be computed; Select[i] = false otherwise.

5: n – Integer

   Input

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

6: t – const double

   Input

   On entry: \( T \), the upper quasi-triangular matrix.

7: pdt – Integer

   Input

   On entry: the first dimension of the array \( t \).

8: vl – double

   Input/Output

   On entry: if how_many = Nag_Selected and select[i] = true, vl[i] contains the \( i \)th selected eigenvalue.

9: pdvl – Integer

   Input

   On entry: the first dimension of the array \( vl \).

10: vr – double

    Input/Output

    On entry: if how_many = Nag_Selected and select[i] = true, vr[i] contains the \( i \)th selected eigenvalue.

11: pdvr – Integer

    Input

    On entry: the first dimension of the array \( vr \).

12: mm – Integer

    Input

    On entry: the number of columns in \( vl \) and \( vr \).

13: m – Integer

    Output

    On exit: \( m \), the number of eigenvectors actually computed.

14: fail – NagError

    Output

    On exit: information about any error conditions.

[NP3645/7]
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

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, 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, `select[j]` must be set `TRUE`. To select the complex eigenvector corresponding to a complex conjugate pair of eigenvalues `λ_i` and `λ_{i+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: `n ≥ 0`.

6: `t[dim]` – const double

Input

Note: the dimension, `dim`, of the array `t` must be at least `max(1, pdt × n)`. If `order = Nag_ColMajor`, the `(i, j)`th element of the matrix `T` is stored in `t[(j - 1) × pdt + i - 1]` and if `order = Nag_RowMajor`, the `(i, j)`th element of the matrix `T` is stored in `t[(i - 1) × 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: `pdt ≥ max(1, n)`.

8: `vl[dim]` – double

Input/Output

Note: the dimension, `dim`, of the array `vl` must be at least
- `max(1, pdvl × mm)` when `side = Nag_LeftSide` or `Nag_BothSides` and `order = Nag_ColMajor`;
- `max(1, pdvl × 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 `vl[(j - 1) × pdvl + i - 1]` and if `order = Nag_RowMajor`, the `(i, j)`th element of the matrix is stored in `vl[(i - 1) × 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

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 \( \geq \) \( \max(1, n) \);
  if side = Nag_RightSide, pdvl \( \geq 1 \);

if order = Nag_RowMajor,
  if side = Nag_LeftSide or Nag_BothSides, pdvl \( \geq \) \( \max(1, mm) \);
  if side = Nag_RightSide, pdvl \( \geq 1 \).

10: \( \text{vr[dim]} \) – double

Input/Output

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

\[
\max(1, \text{pdvr} \times \text{mm}) \quad \text{when} \quad \text{side} = \text{Nag_RightSide} \quad \text{or} \quad \text{Nag_BothSides} \quad \text{and} \quad \text{order} = \text{Nag_ColMajor};
\]

\[
\max(1, \text{pdvr} \times n) \quad \text{when} \quad \text{side} = \text{Nag_RightSide} \quad \text{or} \quad \text{Nag_BothSides} \quad \text{and} \quad \text{order} = \text{Nag_RowMajor};
\]

1 \quad \text{when} \quad \text{side} = \text{Nag_LeftSide}.

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

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 \( \geq \) \( \max(1, n) \);
  if side = Nag_LeftSide, pdvr \( \geq 1 \);

if order = Nag_RowMajor,
  if side = Nag_RightSide or Nag_BothSides, pdvr \( \geq \) \( \max(1, mm) \);
if side = Nag_LeftSide, pdvr ≥ 1.

12: \( \text{mm} \) – Integer \( \text{Input} \)

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

\( \text{Constraint: mm} \geq \text{required_rowcol} \).

13: \( \text{m} \) – Integer * \( \text{Output} \)

On exit: \( \text{required_rowcol} \), the number of rows or columns of \( \text{vl} \) and/or \( \text{vr} \) actually used to store the computed eigenvectors. If \( \text{how_many} = \text{Nag_ComputeAll} \) or \( \text{Nag_BackTransform} \), \( \text{m} \) is set to \( n \).

14: \( \text{fail} \) – NagError * \( \text{Output} \)

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, \( n = \langle \text{value} \rangle \).

Constraint: \( n \geq 0 \).

On entry, \( \text{mm} = \langle \text{value} \rangle \).

Constraint: \( \text{mm} \geq \text{required_rowcol} \), where \( \text{required_rowcol} \) is obtained by counting 1 for each selected real eigenvector and 2 for each selected complex eigenvector.

On entry, \( \text{pdt} = \langle \text{value} \rangle \).

Constraint: \( \text{pdt} > 0 \).

On entry, \( \text{pdvl} = \langle \text{value} \rangle \).

Constraint: \( \text{pdvl} > 0 \).

On entry, \( \text{pdvr} = \langle \text{value} \rangle \).

Constraint: \( \text{pdvr} > 0 \).

NE_INT_2

On entry, \( \text{pdt} = \langle \text{value} \rangle \), \( n = \langle \text{value} \rangle \).

Constraint: \( \text{pdt} \geq \max(1, n) \).

NE_ENUM_INT_2

On entry, \( \text{side} = \langle \text{value} \rangle \), \( n = \langle \text{value} \rangle \), \( \text{pdvl} = \langle \text{value} \rangle \).

Constraint: if \( \text{side} = \text{Nag_LeftSide} \) or \( \text{Nag_BothSides} \), \( \text{pdvl} \geq \max(1, n) \); if \( \text{side} = \text{Nag_RightSide} \), \( \text{pdvl} \geq 1 \).

On entry, \( \text{side} = \langle \text{value} \rangle \), \( n = \langle \text{value} \rangle \), \( \text{pdvr} = \langle \text{value} \rangle \).

Constraint: if \( \text{side} = \text{Nag_RightSide} \) or \( \text{Nag_BothSides} \), \( \text{pdvr} \geq \max(1, n) \); if \( \text{side} = \text{Nag_LeftSide} \), \( \text{pdvr} \geq 1 \).

On entry, \( \text{side} = \langle \text{value} \rangle \), \( \text{mm} = \langle \text{value} \rangle \), \( \text{pdvl} = \langle \text{value} \rangle \).

Constraint: if \( \text{side} = \text{Nag_LeftSide} \) or \( \text{Nag_BothSides} \), \( \text{pdvl} \geq \max(1, \text{mm}) \); if \( \text{side} = \text{Nag_RightSide} \), \( \text{pdvl} \geq 1 \).

On entry, \( \text{side} = \langle \text{value} \rangle \), \( \text{mm} = \langle \text{value} \rangle \), \( \text{pdvr} = \langle \text{value} \rangle \).

Constraint: if \( \text{side} = \text{Nag_RightSide} \) or \( \text{Nag_BothSides} \), \( \text{pdvr} \geq \max(1, \text{mm}) \); if \( \text{side} = \text{Nag_LeftSide} \), \( \text{pdvr} \geq 1 \).
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:

\[
\theta(\tilde{x}_i, x_i) \leq \frac{c(a)c\|T\|_2}{\text{sep}_i}
\]

where \( \text{sep}_i \) is the reciprocal condition number of \( x_i \).

The condition number \( \text{sep}_i \) 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).