1 Purpose

nag_ztrevc (f08qxc) computes selected left and/or right eigenvectors of a complex upper triangular matrix.

2 Specification

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

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

3 Description

nag_ztrevc (f08qxc) computes left and/or right eigenvectors of a complex upper triangular matrix $T$. Such a matrix arises from the Schur factorization of a complex general matrix, as computed by nag_zhseqr (f08psc), for example.

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

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

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 a unitary matrix from the Schur factorization of a matrix $A$ as $A = QTQ^H$; 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 $\max(|\text{Re}(x_i)| + |\text{Im}(x_i)|) = 1$.

4 References


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 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:  
   **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|] – const Nag_Boolean Input

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 eigenvector corresponding to the eigenvalue \( \lambda_j \), select[|j| - 1] must be set
    Nag_TRUE.

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|] – Complex Input/Output

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 triangular matrix T, as returned by nag_zhseqr (f08psc).
On exit: is used as internal workspace prior to being restored and hence is unchanged.

7: pdt – Integer Input

On entry: the stride separating row or column elements (depending on the value of order) in the
    array t.
Constraints:
    if order = Nag_ColMajor, pdt \geq \max(1, n);
    if order = Nag_RowMajor, pdt \geq n.
8: \( \text{vl}[\text{dim}] \) – Complex

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

- \( \text{pdvl} \times \text{mm} \) when \( \text{side} = \text{Nag\_LeftSide} \) or \( \text{Nag\_BothSides} \) and \( \text{order} = \text{Nag\_ColMajor} \);
- \( n \times \text{pdvl} \) when \( \text{side} = \text{Nag\_LeftSide} \) or \( \text{Nag\_BothSides} \) and \( \text{order} = \text{Nag\_RowMajor} \);
- otherwise \( \text{vl} \) may be \text{NULL}.

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

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

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

**On entry:** if \( \text{how\_many} = \text{Nag\_BackTransform} \) and \( \text{side} = \text{Nag\_LeftSide} \) or \( \text{Nag\_BothSides} \), \( \text{vl} \) must contain an \( n \) by \( n \) matrix \( Q \) (usually the matrix of Schur vectors returned by nag\_zhseqr (f08psc)).

If \( \text{how\_many} = \text{Nag\_ComputeAll} \) or \( \text{Nag\_ComputeSelected} \), \( \text{vl} \) need not be set.

**On exit:** if \( \text{side} = \text{Nag\_LeftSide} \) or \( \text{Nag\_BothSides} \), \( \text{vl} \) contains the computed left eigenvectors (as specified by \( \text{how\_many} \) and \( \text{select} \)). The eigenvectors are stored consecutively in the rows or columns (depending on the value of \( \text{order} \)) of the array, in the same order as their eigenvalues.

If \( \text{side} = \text{Nag\_RightSide} \), \( \text{vl} \) is not referenced and may be \text{NULL}.

9: \( \text{pdvl} \) – Integer

**Input**

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

**Constraints:**

- if \( \text{order} = \text{Nag\_ColMajor} \),
  - if \( \text{side} = \text{Nag\_LeftSide} \) or \( \text{Nag\_BothSides} \), \( \text{pdvl} \geq n \);
  - if \( \text{side} = \text{Nag\_RightSide} \), \( \text{vl} \) may be \text{NULL}.
- if \( \text{order} = \text{Nag\_RowMajor} \),
  - if \( \text{side} = \text{Nag\_LeftSide} \) or \( \text{Nag\_BothSides} \), \( \text{pdvl} \geq \text{mm} \);
  - if \( \text{side} = \text{Nag\_RightSide} \), \( \text{vl} \) may be \text{NULL}.

10: \( \text{vr}[\text{dim}] \) – Complex

**Input/Output**

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

- \( \text{pdvr} \times \text{mm} \) when \( \text{side} = \text{Nag\_RightSide} \) or \( \text{Nag\_BothSides} \) and \( \text{order} = \text{Nag\_ColMajor} \);
- \( n \times \text{pdvr} \) when \( \text{side} = \text{Nag\_RightSide} \) or \( \text{Nag\_BothSides} \) and \( \text{order} = \text{Nag\_RowMajor} \);
- otherwise \( \text{vr} \) may be \text{NULL}.

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

\[
\text{vr}[(j - 1) \times \text{pdvr} + i - 1] \quad \text{when } \text{order} = \text{Nag\_ColMajor};
\]

\[
\text{vr}[(i - 1) \times \text{pdvr} + j - 1] \quad \text{when } \text{order} = \text{Nag\_RowMajor}.
\]

**On entry:** if \( \text{how\_many} = \text{Nag\_BackTransform} \) and \( \text{side} = \text{Nag\_RightSide} \) or \( \text{Nag\_BothSides} \), \( \text{vr} \) must contain an \( n \) by \( n \) matrix \( Q \) (usually the matrix of Schur vectors returned by nag\_zhseqr (f08psc)).

If \( \text{how\_many} = \text{Nag\_ComputeAll} \) or \( \text{Nag\_ComputeSelected} \), \( \text{vr} \) need not be set.

**On exit:** if \( \text{side} = \text{Nag\_RightSide} \) or \( \text{Nag\_BothSides} \), \( \text{vr} \) contains the computed right eigenvectors (as specified by \( \text{how\_many} \) and \( \text{select} \)). The eigenvectors are stored consecutively in the rows or columns (depending on the value of \( \text{order} \)) of the array, in the same order as their eigenvalues.

If \( \text{side} = \text{Nag\_LeftSide} \), \( \text{vr} \) is not referenced and may be \text{NULL}.
11: \textbf{pdvr} – Integer \hfill \textit{Input}

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

\textit{Constraints:}

\begin{itemize}
  \item if \textbf{order} = Nag\_ColMajor,
    \begin{itemize}
      \item if \textbf{side} = Nag\_RightSide or Nag\_BothSides, \textbf{pdvr} \geq n;
      \item if \textbf{side} = Nag\_LeftSide, \textbf{vr} may be NULL.;
    \end{itemize}
  \item if \textbf{order} = Nag\_RowMajor,
    \begin{itemize}
      \item if \textbf{side} = Nag\_RightSide or Nag\_BothSides, \textbf{pdvr} \geq \textbf{mm};
      \item if \textbf{side} = Nag\_LeftSide, \textbf{vr} may be NULL.;
    \end{itemize}
\end{itemize}

12: \textbf{mm} – Integer \hfill \textit{Input}

\textit{On entry:} the number of rows or columns (depending on the value of \textbf{order}) in the arrays \textbf{vl} and/or \textbf{vr}. The precise number of rows or columns required, \textit{required}, is \textit{n} if \textbf{how\_many} = Nag\_ComputeAll or Nag\_BackTransform; if \textbf{how\_many} = Nag\_ComputeSelected, \textit{required} is the number of selected eigenvectors (see \textbf{select}), in which case \textit{0 \leq required \leq n}.

\textit{Constraints:}

\begin{itemize}
  \item if \textbf{how\_many} = Nag\_ComputeAll or Nag\_BackTransform, \textbf{mm} \geq \textit{n};
  \item otherwise \textbf{mm} \geq \textit{required}.
\end{itemize}

13: \textbf{m} – Integer \hfill \textit{Output}

\textit{On exit:} \textit{required}, the number of selected eigenvectors. If \textbf{how\_many} = Nag\_ComputeAll or Nag\_BackTransform, \textbf{m} is set to \textit{n}.

14: \textbf{fail} – Nag\_Error\* \hfill \textit{Input/Output}

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

6 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}

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

\textbf{NE\_ENUM\_INT\_2}

On entry, \textbf{how\_many} = \langle\textit{value}\rangle, \textbf{mm} = \langle\textit{value}\rangle and \textbf{n} = \langle\textit{value}\rangle. Constraint: if \textbf{how\_many} = Nag\_ComputeAll or Nag\_BackTransform, \textbf{mm} \geq \textit{n}; otherwise \textbf{mm} \geq \textit{required}.

On entry, \textbf{side} = \langle\textit{value}\rangle, \textbf{pdvl} = \langle\textit{value}\rangle, \textbf{mm} = \langle\textit{value}\rangle. Constraint: if \textbf{side} = Nag\_LeftSide or Nag\_BothSides, \textbf{pdvl} \geq \textbf{mm}.

On entry, \textbf{side} = \langle\textit{value}\rangle, \textbf{pdvl} = \langle\textit{value}\rangle and \textbf{n} = \langle\textit{value}\rangle. Constraint: if \textbf{side} = Nag\_LeftSide or Nag\_BothSides, \textbf{pdvl} \geq \textbf{n}.
On entry, side = (value), pdvra = (value), mm = (value).
Constraint: if side = Nag_RightSide or Nag_BothSides, pdvra ≥ mm.
On entry, side = (value), pdvra = (value) and n = (value).
Constraint: if side = Nag_RightSide or Nag_BothSides, pdvra ≥ n.

NE_INT
On entry, n = (value).
Constraint: n > 0.
On entry, n = (value).
Constraint: n ≥ 0.
On entry, pdt = (value).
Constraint: pdt > 0.
On entry, pdvra = (value).
Constraint: pdvra > 0.
On entry, pdvra = (value).
Constraint: pdvra > 0.

NE_INT_2
On entry, pdt = (value) and n = (value).
Constraint: pdt ≥ max(1, n).
On entry, pdt = (value) and n = (value).
Constraint: pdt ≥ 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
If xi is an exact right eigenvector, and x̂ i is the corresponding computed eigenvector, then the angle θ(·x̂ i, xi) between them is bounded as follows:

\[ \theta(\hat{x}_i, x_i) \leq \frac{c(n)c\|T\|_2}{sep_i} \]

where sep i is the reciprocal condition number of xi.
The condition number sep i may be computed by calling nag_ztrsna (f08qyc).

8 Parallelism and Performance
nag_ztrevc (f08qxc) is not threaded by NAG in any implementation.

nag_ztrevc (f08qxc) 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

The real analogue of this function is nag_dtrevc (f08qkc).

10 Example

See Section 10 in nag_zgebal (f08nvc).