NAG Library Function Document

nag_zgghrd (f08wsc)

1 Purpose

nag_zgghrd (f08wsc) reduces a pair of complex matrices \((A, B)\), where \(B\) is upper triangular, to the generalized upper Hessenberg form using unitary transformations.

2 Specification

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

void nag_zgghrd (Nag_OrderType order, Nag_ComputeQType compq,
                Nag_ComputeZType compz, Integer n, Integer ilo, Integer ihi,
                Complex a[], Integer pda, Complex b[], Integer pdb, Complex q[],
                Integer pdq, Complex z[], Integer pdz, NagError *fail)
```

3 Description

nag_zgghrd (f08wsc) is usually the third step in the solution of the complex generalized eigenvalue problem

\[
Ax = \lambda Bx.
\]

The (optional) first step balances the two matrices using nag_zggbal (f08wvc). In the second step, matrix \(B\) is reduced to upper triangular form using the \(QR\) factorization function nag_zgeqrf (f08asc) and this unitary transformation \(Q\) is applied to matrix \(A\) by calling nag_zunmqr (f08auc).

nag_zgghrd (f08wsc) reduces a pair of complex matrices \((A, B)\), where \(B\) is triangular, to the generalized upper Hessenberg form using unitary transformations. This two-sided transformation is of the form

\[
Q^H AZ = H
\]
\[
Q^H BZ = T
\]

where \(H\) is an upper Hessenberg matrix, \(T\) is an upper triangular matrix and \(Q\) and \(Z\) are unitary matrices determined as products of Givens rotations. They may either be formed explicitly, or they may be postmultiplied into input matrices \(Q_1\) and \(Z_1\), so that

\[
Q_1 AZ_1^H = (Q_1Q)H(Z_1Z)^H,
\]
\[
Q_1 BZ_1^H = (Q_1Q)T(Z_1Z)^H.
\]

4 References


5 Arguments

1:  \textbf{order} – Nag_OrderType

\textit{Input}

\textit{On entry}: the \texttt{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
**f08wsc**

**NAG Library Manual**

**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: \[ \text{compq} - \text{Nag\_ComputeQType} \quad \text{Input} \]

*On entry:* specifies the form of the computed unitary matrix \( Q \).

- \( \text{compq} = \text{Nag\_NotQ} \)
  
  Do not compute \( Q \).

- \( \text{compq} = \text{Nag\_InitQ} \)
  
  The unitary matrix \( Q \) is returned.

- \( \text{compq} = \text{Nag\_UpdateSchur} \)
  
  \( q \) must contain a unitary matrix \( Q_1 \), and the product \( Q_1Q \) is returned.

*Constraint: compq = Nag\_NotQ, Nag\_InitQ or Nag\_UpdateSchur.*

3: \[ \text{compz} - \text{Nag\_ComputeZType} \quad \text{Input} \]

*On entry:* specifies the form of the computed unitary matrix \( Z \).

- \( \text{compz} = \text{Nag\_NotZ} \)
  
  Do not compute \( Z \).

- \( \text{compz} = \text{Nag\_UpdateZ} \)
  
  \( z \) must contain a unitary matrix \( Z_1 \), and the product \( Z_1Z \) is returned.

- \( \text{compz} = \text{Nag\_InitZ} \)
  
  The unitary matrix \( Z \) is returned.

*Constraint: compz = Nag\_NotZ, Nag\_UpdateZ or Nag\_InitZ.*

4: \[ \text{n} - \text{Integer} \quad \text{Input} \]

*On entry:* \( n \), the order of the matrices \( A \) and \( B \).

*Constraint: n \geq 0.*

5: \[ \text{ilo} - \text{Integer} \quad \text{Input} \]

6: \[ \text{ihi} - \text{Integer} \quad \text{Input} \]

*On entry:* \( i_{lo} \) and \( i_{hi} \) as determined by a previous call to nag_zggbal (f08wvc). Otherwise, they should be set to 1 and \( n \), respectively.

*Constraints:*

- if \( n > 0 \), \( 1 \leq \text{ilo} \leq \text{ihi} \leq n \);
- if \( n = 0 \), \( \text{ilo} = 1 \) and \( \text{ihi} = 0 \).

7: \[ \text{a}[\text{dim}] - \text{Complex} \quad \text{Input/Output} \]

*Note:* the dimension, \( \text{dim} \), of the array \( a \) must be at least \( \max(1, \text{pda} \times n) \).

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

- \( a[(j-1) \times \text{pda} + i - 1] \) when \( \text{order} = \text{Nag\_ColMajor} \);
- \( a[(i-1) \times \text{pda} + j - 1] \) when \( \text{order} = \text{Nag\_RowMajor} \).

*On entry:* the matrix \( A \) of the matrix pair \((A, B)\). Usually, this is the matrix \( A \) returned by nag_zunmqr (f08auc).

*On exit:* \( a \) is overwritten by the upper Hessenberg matrix \( H \).
8: \textbf{pda} – Integer \hfill \textit{Input}

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

\textit{Constraint}: \texttt{pda} ≥ \max(1, \texttt{n}).

9: \textbf{b[dim]} – Complex \hfill \textit{Input/Output}

\textbf{Note}: the dimension, \texttt{dim}, of the array \texttt{b} must be at least \max(1, \texttt{pdb} \times \texttt{n}).

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

\[ b[(j - 1) \times \texttt{pdb} + i - 1] \text{ when } \texttt{order} = \texttt{Nag\_ColMajor}; \]
\[ b[(i - 1) \times \texttt{pdb} + j - 1] \text{ when } \texttt{order} = \texttt{Nag\_RowMajor}. \]

\textit{On entry}: the upper triangular matrix \(B\) of the matrix pair \((A, B)\). Usually, this is the matrix \(B\) returned by the \(QR\) factorization function \texttt{nag\_zgeqrf} (f08asc).

\textit{On exit}: \texttt{b} is overwritten by the upper triangular matrix \(T\).

10: \textbf{pdb} – Integer \hfill \textit{Input}

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

\textit{Constraint}: \texttt{pdb} ≥ \max(1, \texttt{n}).

11: \textbf{q[dim]} – Complex \hfill \textit{Input/Output}

\textbf{Note}: the dimension, \texttt{dim}, of the array \texttt{q} must be at least

\[
\begin{align*}
\max(1, \texttt{pdq} \times \texttt{n}) \text{ when } \texttt{compq} = \texttt{Nag\_InitQ} \text{ or } \texttt{Nag\_UpdateSchur}; \\
1 \text{ when } \texttt{compq} = \texttt{Nag\_NotQ}.
\end{align*}
\]

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

\[ q[(j - 1) \times \texttt{pdq} + i - 1] \text{ when } \texttt{order} = \texttt{Nag\_ColMajor}; \]
\[ q[(i - 1) \times \texttt{pdq} + j - 1] \text{ when } \texttt{order} = \texttt{Nag\_RowMajor}. \]

\textit{On entry}: if \texttt{compq} = \texttt{Nag\_UpdateSchur}, \texttt{q} must contain a unitary matrix \(Q_1\).

If \texttt{compq} = \texttt{Nag\_NotQ}, \texttt{q} is not referenced.

\textit{On exit}: if \texttt{compq} = \texttt{Nag\_InitQ}, \texttt{q} contains the unitary matrix \(Q\).

If \texttt{compq} = \texttt{Nag\_UpdateSchur}, \texttt{q} is overwritten by \(Q_1Q\).

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

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

\textit{Constraints}:

\[
\begin{align*}
\text{if } \texttt{compq} = \texttt{Nag\_InitQ} \text{ or } \texttt{Nag\_UpdateSchur}, & \text{ \texttt{pdq} ≥ } \max(1, \texttt{n}); \\
\text{if } \texttt{compq} = \texttt{Nag\_NotQ}, & \text{ \texttt{pdq} ≥ } 1.
\end{align*}
\]

13: \textbf{z[dim]} – Complex \hfill \textit{Input/Output}

\textbf{Note}: the dimension, \texttt{dim}, of the array \texttt{z} must be at least

\[
\begin{align*}
\max(1, \texttt{pdz} \times \texttt{n}) \text{ when } \texttt{compz} = \texttt{Nag\_UpdateZ} \text{ or } \texttt{Nag\_InitZ}; \\
1 \text{ when } \texttt{compz} = \texttt{Nag\_NotZ}.
\end{align*}
\]

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

\[ z[(j - 1) \times \texttt{pdz} + i - 1] \text{ when } \texttt{order} = \texttt{Nag\_ColMajor}; \]
\[ z[(i - 1) \times \texttt{pdz} + j - 1] \text{ when } \texttt{order} = \texttt{Nag\_RowMajor}. \]

\textit{On entry}: if \texttt{compz} = \texttt{Nag\_UpdateZ}, \texttt{z} must contain a unitary matrix \(Z_1\).
If $\text{compz} = \text{Nag\_NotZ}$, $z$ is not referenced.

On exit: if $\text{compz} = \text{Nag\_InitZ}$, $z$ contains the unitary matrix $Z$.

If $\text{compz} = \text{Nag\_UpdateZ}$, $z$ is overwritten by $Z_1 Z$.

14: $\text{pdz}$ – Integer

Input

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

Constraints:

if $\text{compz} = \text{Nag\_UpdateZ}$ or $\text{Nag\_InitZ}$, $\text{pdz} \geq \max(1, n)$;

if $\text{compz} = \text{Nag\_NotZ}$, $\text{pdz} \geq 1$.

15: $\text{fail}$ – NagError*

Input/Output

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

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.

See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM

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

NE_ENUM_INT_2

On entry, $\text{compq} = \langle\text{value}\rangle$, $\text{pdq} = \langle\text{value}\rangle$ and $\text{n} = \langle\text{value}\rangle$.

Constraint: if $\text{compq} = \text{Nag\_InitQ}$ or $\text{Nag\_UpdateSchur}$, $\text{pdq} \geq \max(1, \text{n})$;

if $\text{compq} = \text{Nag\_NotQ}$, $\text{pdq} \geq 1$.

On entry, $\text{compz} = \langle\text{value}\rangle$, $\text{pdz} = \langle\text{value}\rangle$ and $\text{n} = \langle\text{value}\rangle$.

Constraint: if $\text{compz} = \text{Nag\_UpdateZ}$ or $\text{Nag\_InitZ}$, $\text{pdz} \geq \max(1, \text{n})$;

if $\text{compz} = \text{Nag\_NotZ}$, $\text{pdz} \geq 1$.

NE_INT

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

Constraint: $\text{n} \geq 0$.

On entry, $\text{pda} = \langle\text{value}\rangle$.

Constraint: $\text{pda} > 0$.

On entry, $\text{pdb} = \langle\text{value}\rangle$.

Constraint: $\text{pdb} > 0$.

On entry, $\text{pdq} = \langle\text{value}\rangle$.

Constraint: $\text{pdq} > 0$.

On entry, $\text{pdz} = \langle\text{value}\rangle$.

Constraint: $\text{pdz} > 0$.

NE_INT_2

On entry, $\text{pda} = \langle\text{value}\rangle$ and $\text{n} = \langle\text{value}\rangle$.

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

On entry, $\text{pdb} = \langle\text{value}\rangle$ and $\text{n} = \langle\text{value}\rangle$.

Constraint: $\text{pdb} \geq \max(1, \text{n})$. 
NE_INT_3

On entry, \( n = \langle \text{value} \rangle \), \( \text{ilo} = \langle \text{value} \rangle \) and \( \text{ihi} = \langle \text{value} \rangle \).
Constraint: if \( n > 0 \), \( 1 \leq \text{ilo} \leq \text{ihi} \leq n \);
if \( n = 0 \), \( \text{ilo} = 1 \) and \( \text{ihi} = 0 \).

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

The reduction to the generalized Hessenberg form is implemented using unitary transformations which are backward stable.

8 Parallelism and Performance

Not applicable.

9 Further Comments

This function is usually followed by \texttt{na g\_zhgeqz (f08xsc)} which implements the \( QZ \) algorithm for computing generalized eigenvalues of a reduced pair of matrices.
The real analogue of this function is \texttt{na g\_dgghrd (f08wec)}.

10 Example

See Section 10 in \texttt{na g\_zhgeqz (f08xsc)} and \texttt{na g\_ztgevc (f08yxc)}.

\textit{Mark 25}