NAG C 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
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 post multiplied into input matrices \(Q_1\) and \(Z_1\), so that

\[
Q_1 AZ_1^H = (Q_1 Q) H(Z_1 Z_1)^H, \\
Q_1 BZ_1^H = (Q_1 Q) T(Z_1 Z_1)^H.
\]

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: compq – Nag_ComputeQType
   
   On entry: specifies the form of the computed unitary matrix $Q$, as follows:
   
   if $\text{compq} = \text{Nag\_NotQ}$, do not compute $Q$;
   
   if $\text{compq} = \text{Nag\_InitQ}$, the unitary matrix $Q$ is returned;
   
   if $\text{compq} = \text{Nag\_UpdateSchur}$, $Q$ must contain a unitary matrix $Q_1$, and the product $Q_1 Q$ is returned.

   Constraint: $\text{compq} = \text{Nag\_NotQ}$, $\text{Nag\_InitQ}$ or $\text{Nag\_UpdateSchur}$.

3: compz – Nag_ComputeZType
   
   On entry: specifies the form of the computed unitary matrix $Z$, as follows:
   
   if $\text{compz} = \text{Nag\_NotZ}$, do not compute $Z$;
   
   if $\text{compz} = \text{Nag\_InitZ}$, the unitary matrix $Z$ is returned;
   
   if $\text{compz} = \text{Nag\_UpdateZ}$, $Z$ must contain a unitary matrix $Z_1$, and the product $Z_1 Z$ is returned.

   Constraint: $\text{compz} = \text{Nag\_NotZ}$, $\text{Nag\_InitZ}$ or $\text{Nag\_UpdateZ}$.

4: n – Integer
   
   On entry: $n$, the order of the matrices $A$ and $B$.

   Constraint: $n \geq 0$.

5: ilo – Integer

6: ihi – Integer
   
   On entry: $ilo$ and $ihi$ as determined by a previous call to nag_zggbal (f08wvc). Otherwise, they should be set to 1 and $n$, respectively.

   Constraints:
   
   if $n > 0$, $1 \leq ilo \leq ihi \leq n$;
   
   if $n = 0$, $ilo = 1$ and $ihi = 0$.

7: a[$\text{dim}$] – Complex
   
   Input/Output

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

   If $\text{order} = \text{Nag\_ColMajor}$, the $(i,j)$th element of the matrix $A$ is stored in $a[(j - 1) \times pda + i - 1]$ and if $\text{order} = \text{Nag\_RowMajor}$, the $(i,j)$th element of the matrix $A$ is stored in $a[(i - 1) \times pda + j - 1]$.

   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: pda – Integer
   
   Input

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

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

9: b[$\text{dim}$] – Complex
   
   Input/Output

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

   If $\text{order} = \text{Nag\_ColMajor}$, the $(i,j)$th element of the matrix $B$ is stored in $b[(j - 1) \times pdb + i - 1]$ and if $\text{order} = \text{Nag\_RowMajor}$, the $(i,j)$th element of the matrix $B$ is stored in $b[(i - 1) \times pdb + j - 1]$.

   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 nag_zgeqrf (f08asc).
On exit: $b$ is overwritten by the upper triangular matrix $T$.

10: $\text{pdb} – \text{Integer}$  
*Input*  
On entry: the stride separating matrix row or column elements (depending on the value of $\text{order}$) in the array $b$.  
*Constraint:* $\text{pdb} \geq \max(1, n)$.

11: $\text{q}[\text{dim}] – \text{Complex}$  
*Input/Output*  
*Note:* the dimension, $\text{dim}$, of the array $\text{q}$ must be at least  
$
\max(1, \text{pdq} \times n) \text{ when } \text{compq} = \text{Nag_InitQ} \text{ or Nag_UpdateSchur;}$

1  when $\text{compq} = \text{Nag_NotQ}$.  
If $\text{order} = \text{Nag_ColMajor}$, the $(i,j)$th element of the matrix $Q$ is stored in $\text{q}[(j-1) \times \text{pdq} + i - 1]$  
and if $\text{order} = \text{Nag_RowMajor}$, the $(i,j)$th element of the matrix $Q$ is stored in  
$q[(i-1) \times \text{pdq} + j - 1]$.  
On entry: if $\text{compq} = \text{Nag_NotQ}$, $q$ is not referenced; if $\text{compq} = \text{Nag_UpdateSchur}$, $q$ must  
contain a unitary matrix $Q_i$.  
On exit: if $\text{compq} = \text{Nag_InitQ}$, $q$ contains the unitary matrix $Q$; if $\text{compq} = \text{Nag_UpdateSchur}$,  
$q$ is overwritten by $Q_i Q$.  

12: $\text{pdq} – \text{Integer}$  
*Input*  
On entry: the stride separating matrix row or column elements (depending on the value of $\text{order}$) in the array $q$.  
*Constraints:*  
if $\text{compq} = \text{Nag_InitQ}$ or $\text{Nag_UpdateSchur}$, $\text{pdq} \geq \max(1, n)$;  
if $\text{compq} = \text{Nag_NotQ}$, $\text{pdq} \geq 1$.  

13: $\text{z}[\text{dim}] – \text{Complex}$  
*Input/Output*  
*Note:* the dimension, $\text{dim}$, of the array $z$ must be at least  
$
\max(1, \text{pdz} \times n) \text{ when } \text{compz} = \text{Nag_UpdateZ} \text{ or Nag_InitZ;}$

1  when $\text{compz} = \text{Nag_NotZ}$.  
If $\text{order} = \text{Nag_ColMajor}$, the $(i,j)$th element of the matrix $Z$ is stored in $\text{z}[(j-1) \times \text{pdz} + i - 1]$  
and if $\text{order} = \text{Nag_RowMajor}$, the $(i,j)$th element of the matrix $Z$ is stored in $z[(i-1) \times \text{pdz} + j - 1]$.  
On entry: if $\text{compz} = \text{Nag_NotZ}$, $z$ is not referenced; if $\text{compz} = \text{Nag_UpdateZ}$, $z$ must contain a  
unitary matrix $Z_i$.  
On exit: if $\text{compz} = \text{Nag_InitZ}$, $z$ contains the unitary matrix $Z$; if $\text{compz} = \text{Nag_UpdateZ}$, $z$ is  
overwritten by $Z_i Z$.  

14: $\text{pdz} – \text{Integer}$  
*Input*  
On entry: the stride separating matrix 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: $\text{fail} – \text{NagError *}$  
*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, \( pda = \langle \text{value} \rangle \).
Constraint: \( pda > 0 \).

On entry, \( pdb = \langle \text{value} \rangle \).
Constraint: \( pdb > 0 \).

On entry, \( pdq = \langle \text{value} \rangle \).
Constraint: \( pdq > 0 \).

On entry, \( pdz = \langle \text{value} \rangle \).
Constraint: \( pdz > 0 \).

NE_INT_2

On entry, \( pda = \langle \text{value} \rangle, n = \langle \text{value} \rangle \).
Constraint: \( pda \geq \max(1, n) \).

On entry, \( pdb = \langle \text{value} \rangle, n = \langle \text{value} \rangle \).
Constraint: \( pdb \geq \max(1, n) \).

On entry, \( pdq = \langle \text{value} \rangle, n = \langle \text{value} \rangle \).
Constraint: if \( \text{compq} = \text{Nag_InitQ} \) or \( \text{Nag_UpdateSchur} \), \( pdq \geq \max(1, n) \); if \( \text{compq} = \text{Nag_NotQ} \), \( pdq \geq 1 \).

On entry, \( pdz = \langle \text{value} \rangle, n = \langle \text{value} \rangle \).
Constraint: if \( \text{compz} = \text{Nag_UpdateZ} \) or \( \text{Nag_InitZ} \), \( pdz \geq \max(1, n) \); if \( \text{compz} = \text{Nag_NotZ} \), \( pdz \geq 1 \).

NE_INT_3

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

NE_ENUM_INT_2

On entry, \( \text{compq} = \langle \text{value} \rangle, n = \langle \text{value} \rangle, pdq = \langle \text{value} \rangle \).
Constraint: if \( \text{compq} = \text{Nag_InitQ} \) or \( \text{Nag_UpdateSchur} \), \( pdq \geq \max(1, n) \); if \( \text{compq} = \text{Nag_NotQ} \), \( pdq \geq 1 \).

On entry, \( \text{compz} = \langle \text{value} \rangle, n = \langle \text{value} \rangle, pdz = \langle \text{value} \rangle \).
Constraint: if \( \text{compz} = \text{Nag_UpdateZ} \) or \( \text{Nag_InitZ} \), \( pdz \geq \max(1, n) \); if \( \text{compz} = \text{Nag_NotZ} \), \( pdz \geq 1 \).

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter \( \langle \text{value} \rangle \) 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
The reduction to the generalized Hessenberg form is implemented using unitary transformations which are backward stable.

8 Further Comments
This function is usually followed by nag_zhgeqz (f08xsc) which implements the QZ algorithm for computing generalized eigenvalues of a reduced pair of matrices.
The real analogue of this function is nag_dgghrd (f08wec).

9 Example
See Section 9 of the documents for nag_zhgeqz (f08xsc) and nag_ztgevc (f08yxc).