NAG C Library Function Document

nag_dgghrd (f08wec)

1 Purpose

nag_dgghrd (f08wec) reduces a pair of real matrices \((A, B)\), where \(B\) is upper triangular, to the generalized upper Hessenberg form using orthogonal transformations.

2 Specification

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

3 Description

nag_dgghrd (f08wec) is the third step in the solution of the real generalized eigenvalue problem

\[ Ax = \lambda Bx. \]

The (optional) first step balances the two matrices using nag_dgbal (f08whc). In the second step, matrix \(B\) is reduced to upper triangular form using the QR factorization function nag_dgeqrf (f08aec) and this orthogonal transformation \(Q\) is applied to matrix \(A\) by calling nag_dormqr (f08agc).

nag_dgghrd (f08wec) reduces a pair of real matrices \((A, B)\), where \(B\) is upper triangular, to the generalized upper Hessenberg form using orthogonal transformations. This two-sided transformation is of the form

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

where \(H\) is an upper Hessenberg matrix, \(T\) is an upper triangular matrix and \(Q\) and \(Z\) are orthogonal, 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

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

4 References


5 Parameters

1: order – Nag_OrderType

\(\text{Input}\)

\(\text{On entry:}\) the \(\text{order}\) 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 \(\text{order} = \text{Nag_RowMajor}\). See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

\(\text{Constraint:}\) \(\text{order} = \text{Nag_ROW_MAJOR}\) or \(\text{Nag_COL_MAJOR}\).

2: compq – Nag_ComputeQType

\(\text{Input}\)

\(\text{On entry:}\) specifies the form of the computed orthogonal matrix \(Q\), as follows:
if \( \text{compq} = \text{Nag\_NotQ} \), do not compute \( Q \);
if \( \text{compq} = \text{Nag\_InitQ} \), the orthogonal matrix \( Q \) is returned;
if \( \text{compq} = \text{Nag\_UpdateSchur} \), \( Q \) must contain an orthogonal 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: \( \text{compz} \) – Nag\_ComputeZType

*Input*

*On entry:* specifies the form of the computed orthogonal matrix \( Z \), as follows:

if \( \text{compz} = \text{Nag\_NotZ} \), do not compute \( Z \);
if \( \text{compz} = \text{Nag\_InitZ} \), the orthogonal matrix \( Z \) is returned;
if \( \text{compz} = \text{Nag\_UpdateZ} \), \( Z \) must contain an orthogonal 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

*Input*

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

**Constraint:** \( n \geq 0 \).

5: \( \text{ilo} \) – Integer
6: \( \text{ihi} \) – Integer

*Input*

*On entry:* \( i_{lo} \) and \( i_{hi} \) as determined by a previous call to nag_dggbal (f08whc). 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: \( a[\text{dim}] \) – double

*Input/Output*

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

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

*On entry:* the matrix \( A \) of the matrix pair \( (A, B) \). Usually, this is the matrix \( A \) returned by nag_dormqr (f08agc).

*On exit:* \( a \) is overwritten by the upper Hessenberg matrix \( H \).

8: \( \text{pda} \) – Integer

*Input*

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

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

9: \( b[\text{dim}] \) – double

*Input/Output*

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

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

*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_dgeqrf (f08aec).

*On exit:* the array \( b \) is overwritten by the upper triangular matrix \( T \).
10:  

\textbf{pdb} – Integer \hspace{1em} \textit{Input}

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

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

11:  

\textbf{q[\textit{dim}]} – double \hspace{1em} \textit{Input/Output}

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

\[
\max(1, \texttt{pdq} \times \texttt{n}) \quad \text{when} \quad \texttt{compq} = \texttt{Nag_InitQ} \text{ or } \texttt{Nag_UpdateSchur};
\]

\[
1 \quad \text{when} \quad \texttt{compq} = \texttt{Nag_NotQ}.
\]

\textit{If order} = \texttt{Nag_ColMajor}, the \((i,j)\)th element of the matrix \(Q\) is stored in \(\texttt{q}[{(j-1) \times \texttt{pdq} + i - 1}]\)

\textit{and if order} = \texttt{Nag_RowMajor}, the \((i,j)\)th element of the matrix \(Q\) is stored in

\[
\texttt{q}[(i-1) \times \texttt{pdq} + j - 1].
\]

\textit{On entry}: if \texttt{compq} = \texttt{Nag_NotQ}, \texttt{q} is not referenced; if \texttt{compq} = \texttt{Nag_UpdateSchur}, \texttt{q} must contain an orthogonal matrix \(Q_1\).

\textit{On exit}: if \texttt{compq} = \texttt{Nag_InitQ}, \texttt{q} contains the orthogonal matrix \(Q\); if \texttt{compq} = \texttt{Nag_UpdateSchur}, \texttt{q} is overwritten by \(Q_1Q\).

12:  

\textbf{pdq} – Integer \hspace{1em} \textit{Input}

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

\textbf{Constraints}:

\[
\text{if} \quad \texttt{compq} = \texttt{Nag_InitQ} \text{ or } \texttt{Nag_UpdateSchur}, \texttt{pdq} \geq \max(1, \texttt{n});
\]

\[
\text{if} \quad \texttt{compq} = \texttt{Nag_NotQ}, \texttt{pdq} \geq 1.
\]

13:  

\textbf{z[\textit{dim}]} – double \hspace{1em} \textit{Input/Output}

\textbf{Note}: the dimension, \textit{dim}, of the array \texttt{z} must be at least \(\max(1, \texttt{pdz} \times \texttt{n})\) when \texttt{compz} = \texttt{Nag_UpdateZ} or \texttt{Nag_InitZ}.

\textit{If order} = \texttt{Nag_ColMajor}, the \((i,j)\)th element of the matrix \(Z\) is stored in \(\texttt{z}[{(j-1) \times \texttt{pdz} + i - 1}]\)

\textit{and if order} = \texttt{Nag_RowMajor}, the \((i,j)\)th element of the matrix \(Z\) is stored in \(\texttt{z}[(i-1) \times \texttt{pdz} + j - 1]\).

\textit{On entry}: if \texttt{compz} = \texttt{Nag_NotZ}, \texttt{z} is not referenced; if \texttt{compz} = \texttt{Nag_UpdateZ}, \texttt{z} must contain an orthogonal matrix \(Z_1\).

\textit{On exit}: if \texttt{compz} = \texttt{Nag_InitZ}, \texttt{z} contains the orthogonal matrix \(Z\); if \texttt{compz} = \texttt{Nag_UpdateZ}, \texttt{z} is overwritten by \(Z_1Z\).

14:  

\textbf{pdz} – Integer \hspace{1em} \textit{Input}

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

\textbf{Constraints}:

\textit{if order} = \texttt{Nag_ColMajor},

\[
\text{if} \quad \texttt{compz} = \texttt{Nag_UpdateZ} \text{ or } \texttt{Nag_InitZ}, \texttt{pdz} \geq \max(1, \texttt{n});
\]

\[
\text{if} \quad \texttt{compz} = \texttt{Nag_NotZ}, \texttt{pdz} \geq 1;
\]

\textit{if order} = \texttt{Nag_RowMajor},

\[
\text{if} \quad \texttt{compz} = \texttt{Nag_InitZ} \text{ or } \texttt{Nag_UpdateZ}, \texttt{pdz} \geq \max(1, \texttt{n});
\]

\[
\text{if} \quad \texttt{compz} = \texttt{Nag_NotZ}, \texttt{pdz} \geq 1.
\]

15:  

\textbf{fail} – NagError * \hspace{1em} \textit{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 \( \text{compz} = \text{Nag_InitZ} \) or \( \text{Nag_UpdateZ} \), \( pdz \geq \max(1, n) \); if \( \text{compz} = \text{Nag_NotZ} \), \( pdz \geq 1 \).

**NE_INT_3**

On entry, \( n = \langle \text{value} \rangle, ilo = \langle \text{value} \rangle, ihi = \langle \text{value} \rangle \).
Constraint: if \( n > 0 \), \( 1 \leq ilo \leq ihi \leq n \); if \( n = 0 \), \( ilo = 1 \) 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 orthogonal transformations which are backward stable.

8 Further Comments
This function is usually followed by nag_dhgeqz (f08xec) which implements the QZ algorithm for computing generalized eigenvalues of a reduced pair of matrices.
The complex analogue of this function is nag_zgghrd (f08wsc).

9 Example
See Section 9 of the documents for nag_dhgeqz (f08xec) and nag_dtgevc (f08ykc).